FIG.1

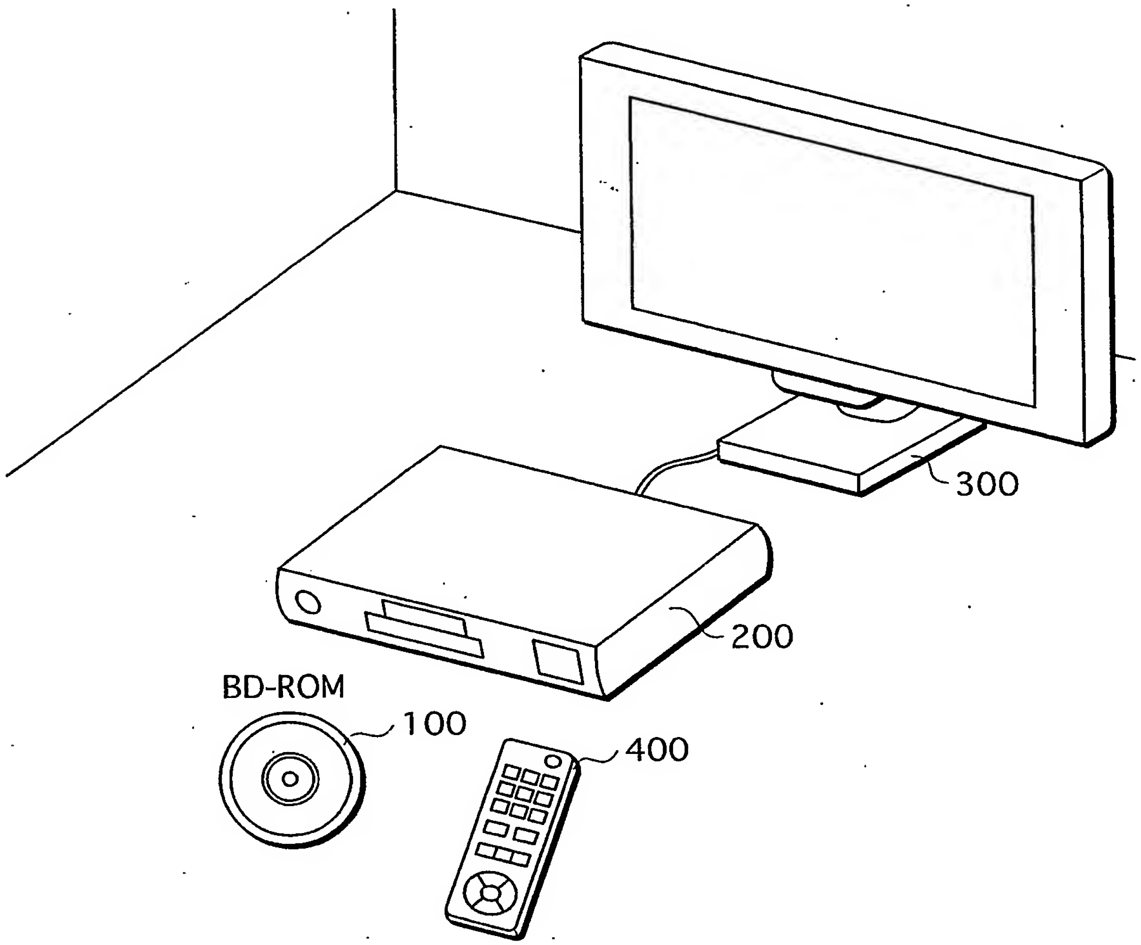


FIG.2

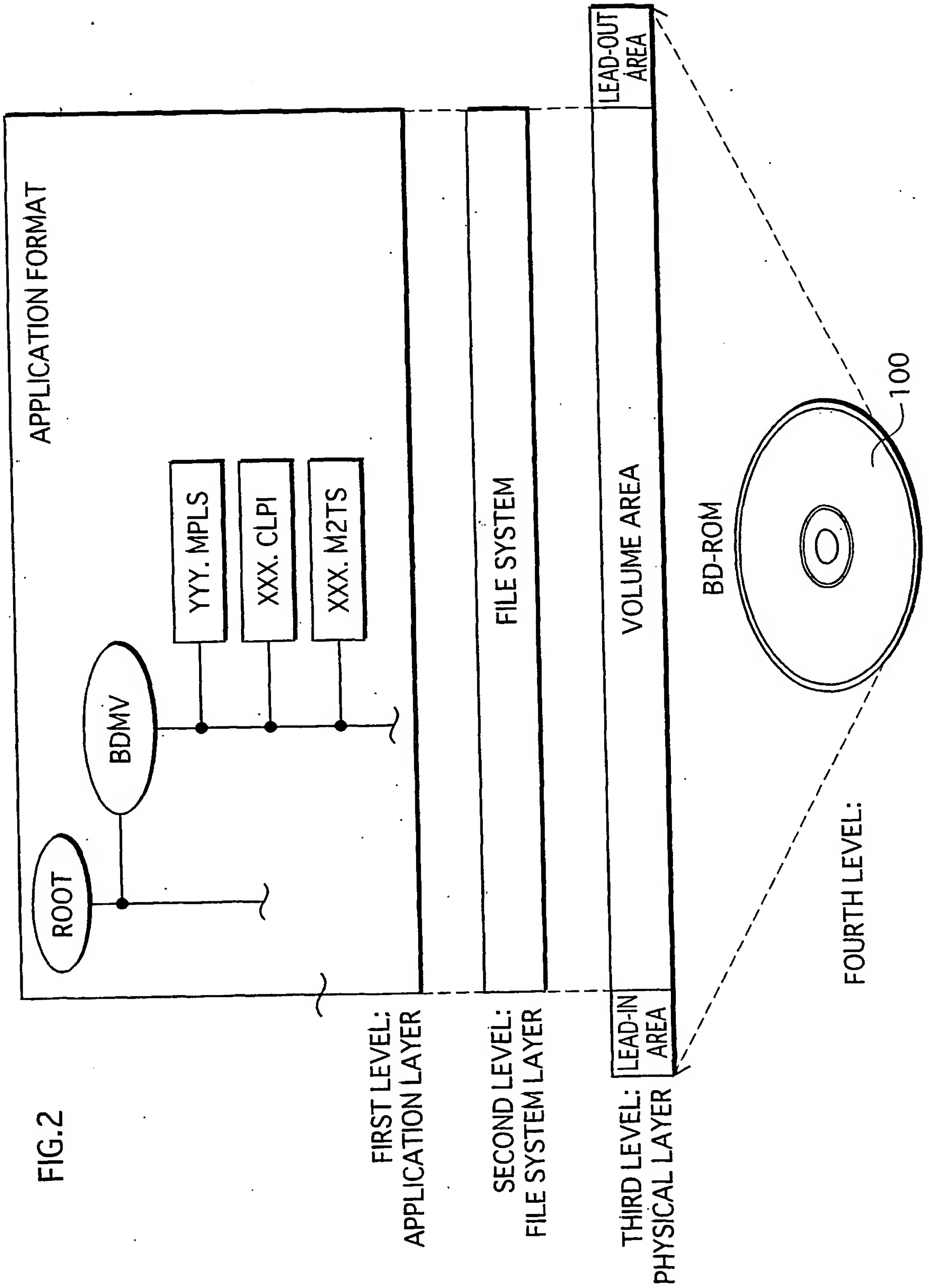

FIRST LEVEL: APPLICATION LAYER
APPLICATION FORMAT
ROOT
BDMV
YYY. MPLS
XXX. CLPI
XXX. M2TS
SECOND LEVEL: FILE SYSTEM LAYER
FILE SYSTEM
THIRD LEVEL: PHYSICAL LAYER
LEAD-IN AREA
VOLUME AREA
LEAD-OUT AREA
FOURTH LEVEL:
BD-ROM
100

FIG.3

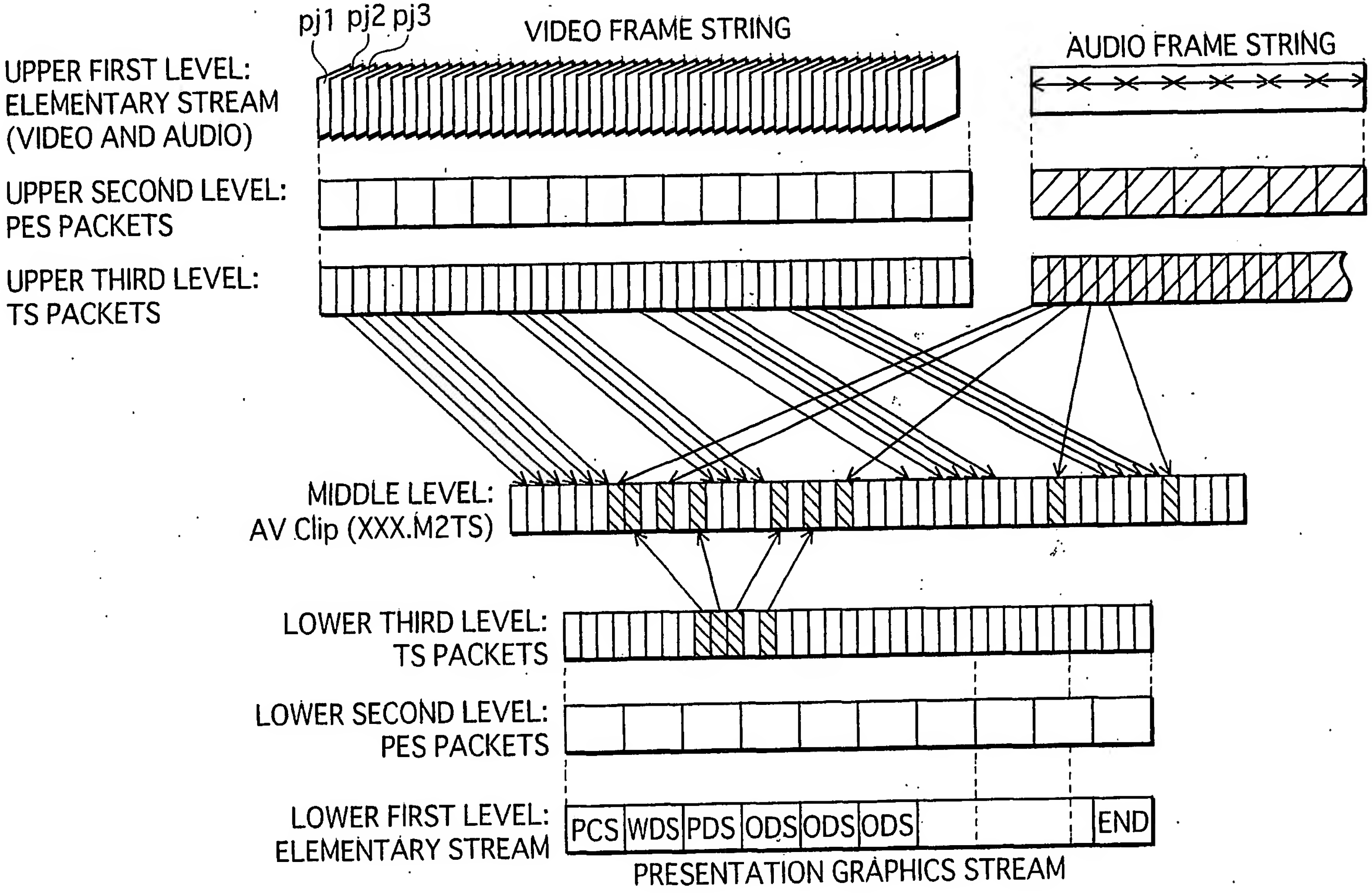

pj1 pj2 pj3
VIDEO FRAME STRING
AUDIO FRAME STRING
UPPER FIRST LEVEL: ELEMENTARY STREAM (VIDEO AND AUDIO)
UPPER SECOND LEVEL: PES PACKETS
UPPER THIRD LEVEL: TS PACKETS
MIDDLE LEVEL: AV Clip (XXX.M2TS)
LOWER THIRD LEVEL: TS PACKETS
LOWER SECOND LEVEL: PES PACKETS
LOWER FIRST LEVEL: ELEMENTARY STREAM
PCS
WDS
PDS
ODS
ODS
ODS
END
PRESENTATION GRAPHICS STREAM

FIG.4A

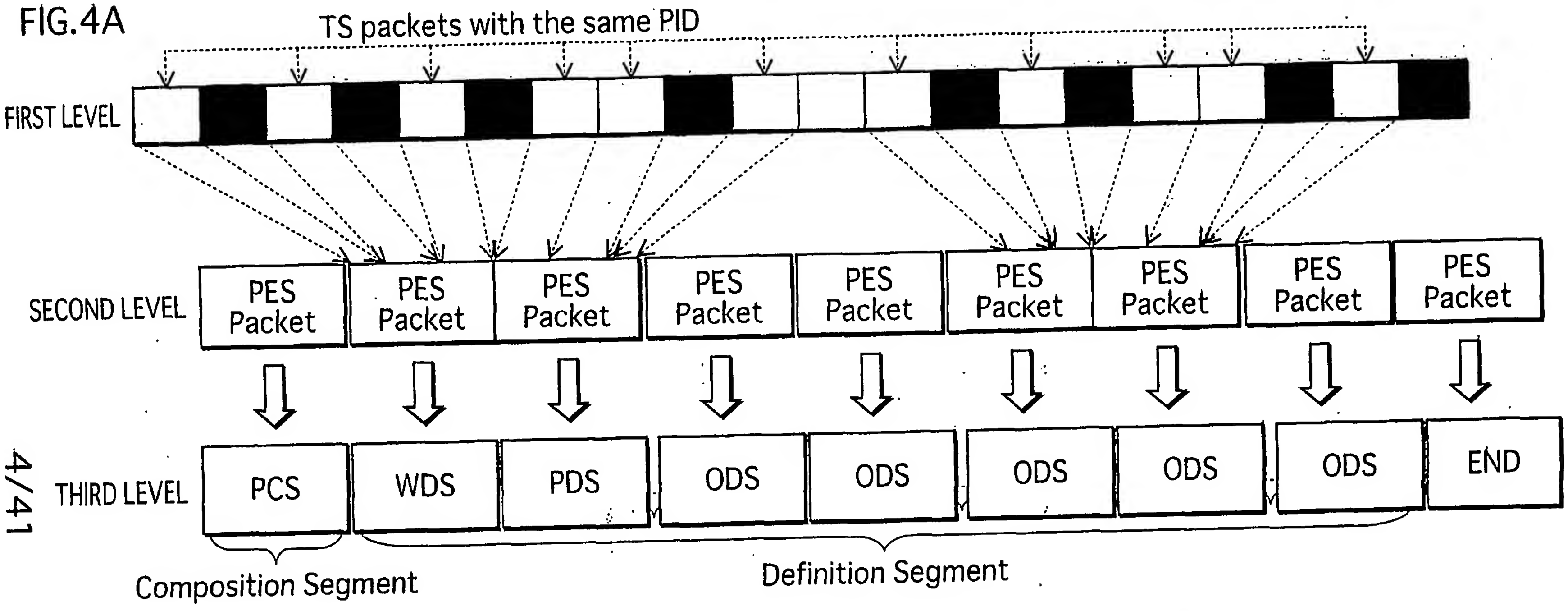

TS packets with the same PID
FIRST LEVEL
SECOND LEVEL
PES Packet
PES Packet
PES Packet
PES Packet
PES Packet
PES Packet
PES Packet
PES Packet
PES Packet
THIRD LEVEL
PCS
WDS
PDS
ODS
ODS
ODS
ODS
ODS
END
Composition Segment
Definition Segment

FIG.4B

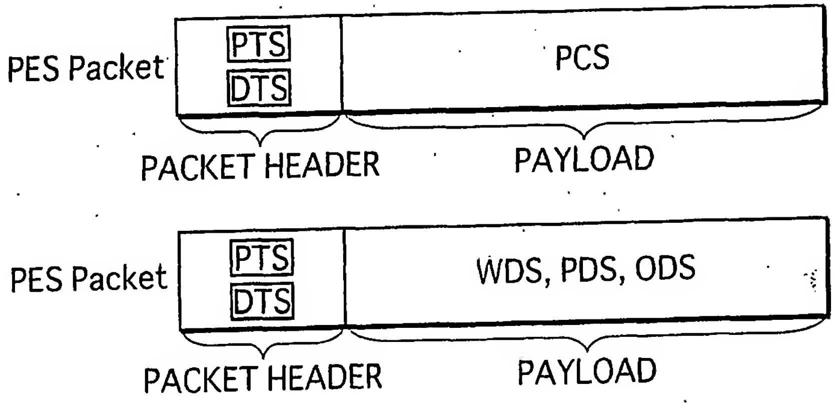

PES Packet
PTS
DTS
PCS
PACKET HEADER
PAYLOAD
PES Packet
PTS
DTS
WDS, PDS, ODS
PACKET HEADER
PAYLOAD

FIG.5

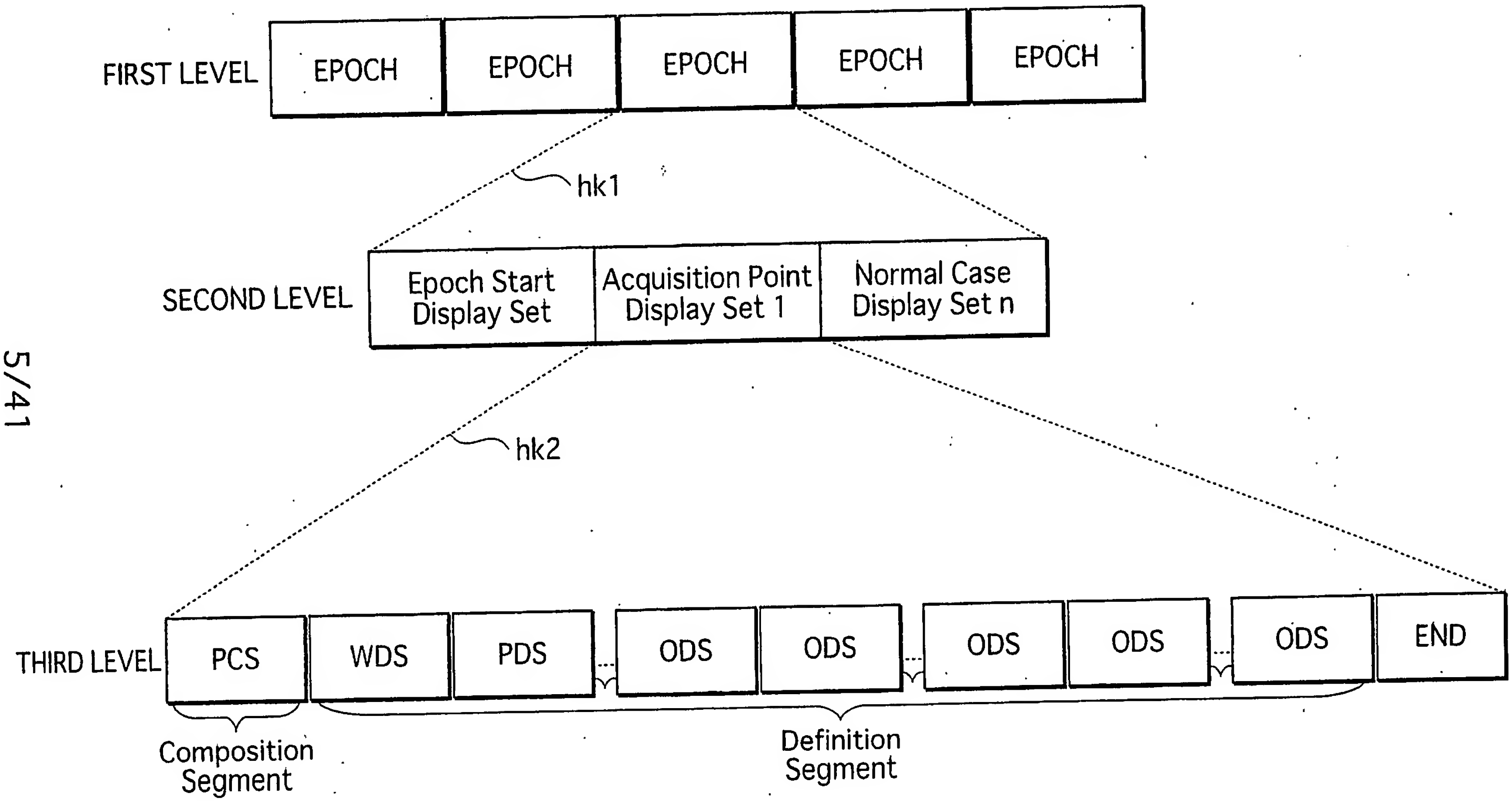

FIRST LEVEL
EPOCH
EPOCH
EPOCH
EPOCH
EPOCH
hk1
SECOND LEVEL
Epoch Start Display Set
Acquisition Point Display Set 1
Normal Case Display Set n
hk2
THIRD LEVEL
PCS
WDS
PDS
ODS
ODS
ODS
ODS
ODS
END
Composition Segment
Definition Segment

FIG.6

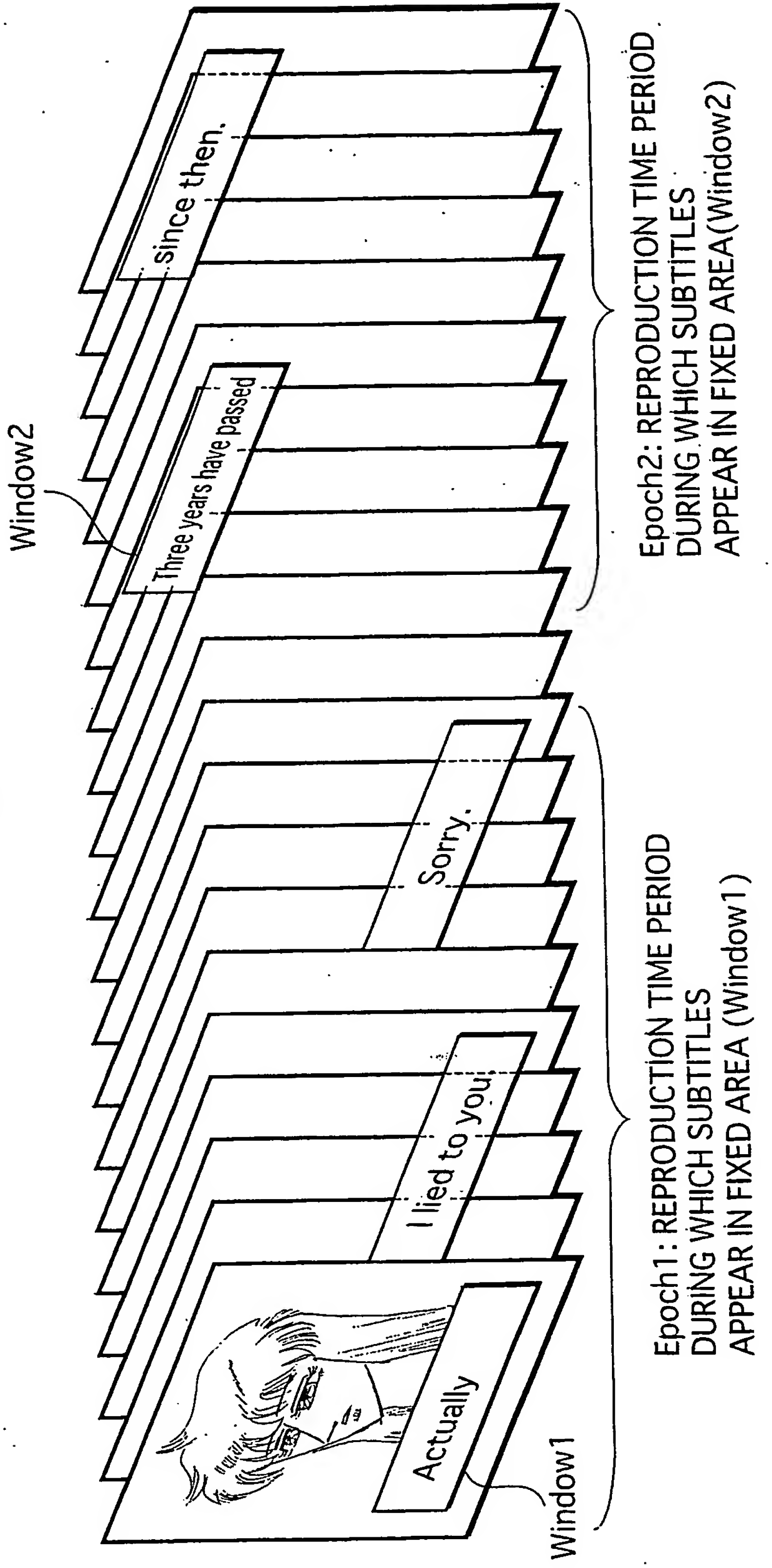

Actually
Window1
I lied to you.
Sorry.
Epoch1: REPRODUCTION TIME PERIOD DURING WHICH SUBTITLES APPEAR IN FIXED AREA (Window1)
Window2
Three years have passed
since then.
Epoch2: REPRODUCTION TIME PERIOD DURING WHICH SUBTITLES APPEAR IN FIXED AREA(Window2)

FIG.7A

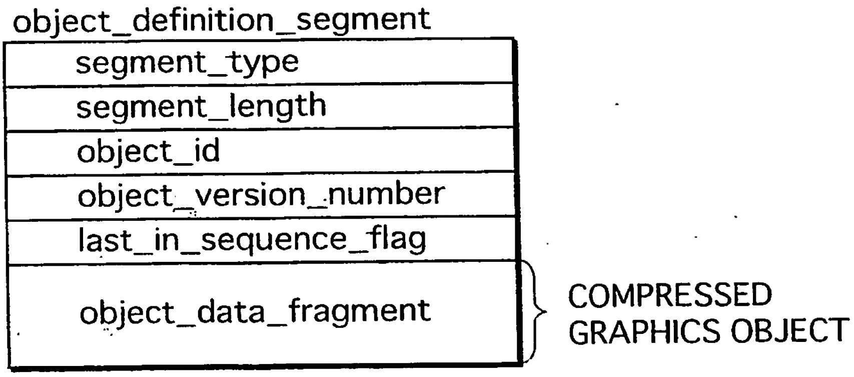

object_definition_segment
segment_type
segment_length
object_id
object_version_number
last_in_sequence_flag
object_data_fragment
COMPRESSED
GRAPHICS OBJECT

FIG.7B

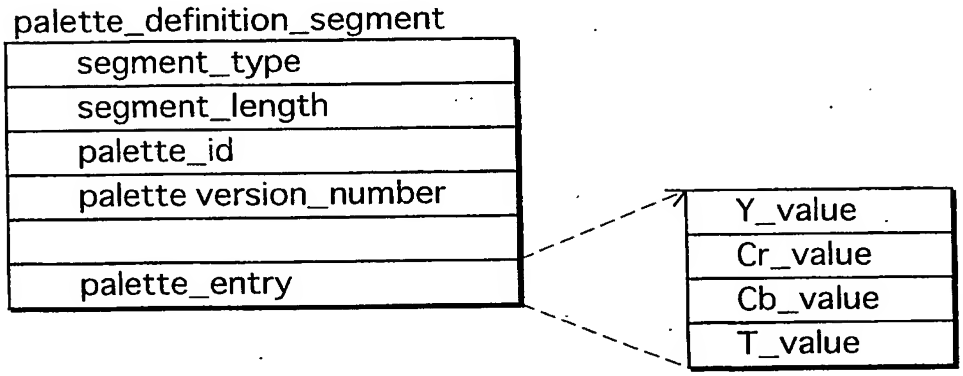

palette_definition_segment
segment_type
segment_length
palette_id
palette version_number
palette_entry
Y_value
Cr_value
Cb_value
T_value

FIG.8A

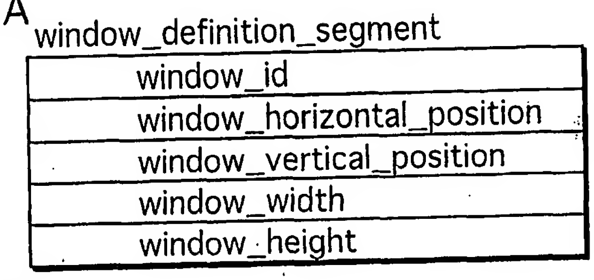

window_definition_segment
window_id
window_horizontal_position
window_vertical_position
window_width
window_height

FIG.8B

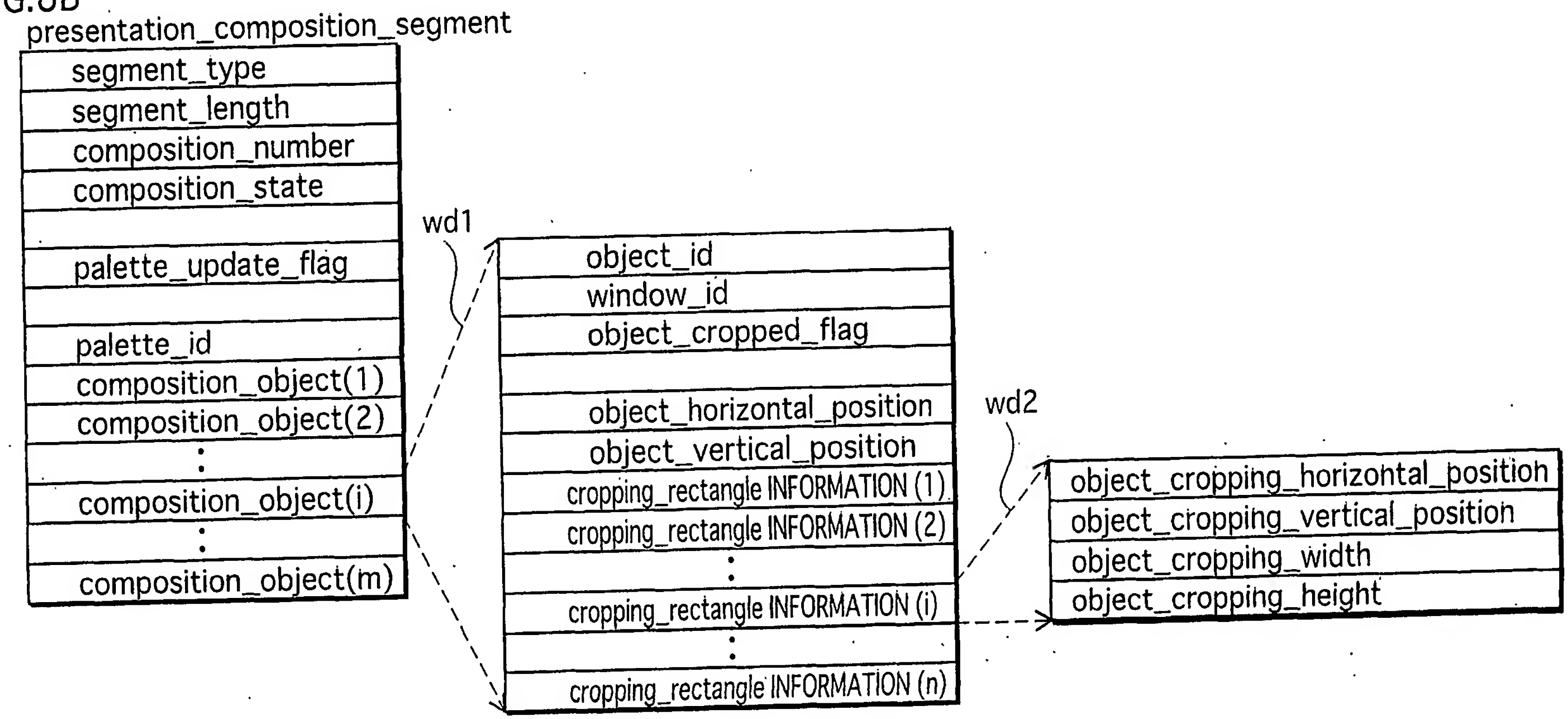

presentation_composition_segment
segment_type
segment_length
composition_number
composition_state
palette_update_flag
palette_id
composition_object(1)
composition_object(2)
composition_object(i)
composition_object(m)
wd1
object_id
window_id
object_cropped_flag
object_horizontal_position
object_vertical_position
cropping_rectangle INFORMATION (1)
cropping_rectangle INFORMATION (2)
cropping_rectangle INFORMATION (i)
cropping_rectangle INFORMATION (n)
wd2
object_cropping_horizontal_position
object_cropping_vertical_position
object_cropping_width
object_cropping_height

FIG.9

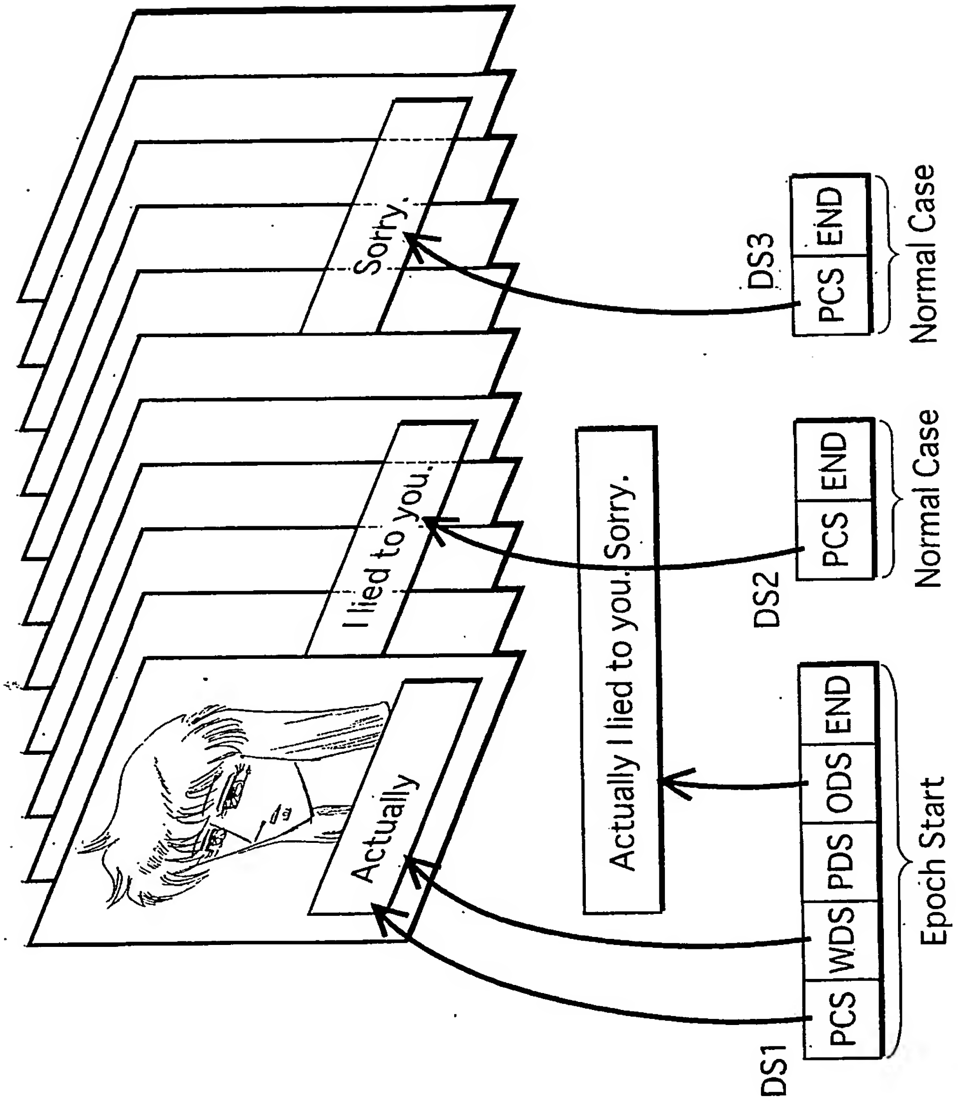

Actually
I lied to you.
Sorry.
Actually I lied to you. Sorry.
DS1
PCS
WDS
PDS
ODS
END
Epoch Start
DS2
PCS
END
Normal Case
DS3
PCS
END
Normal Case

FIG.10

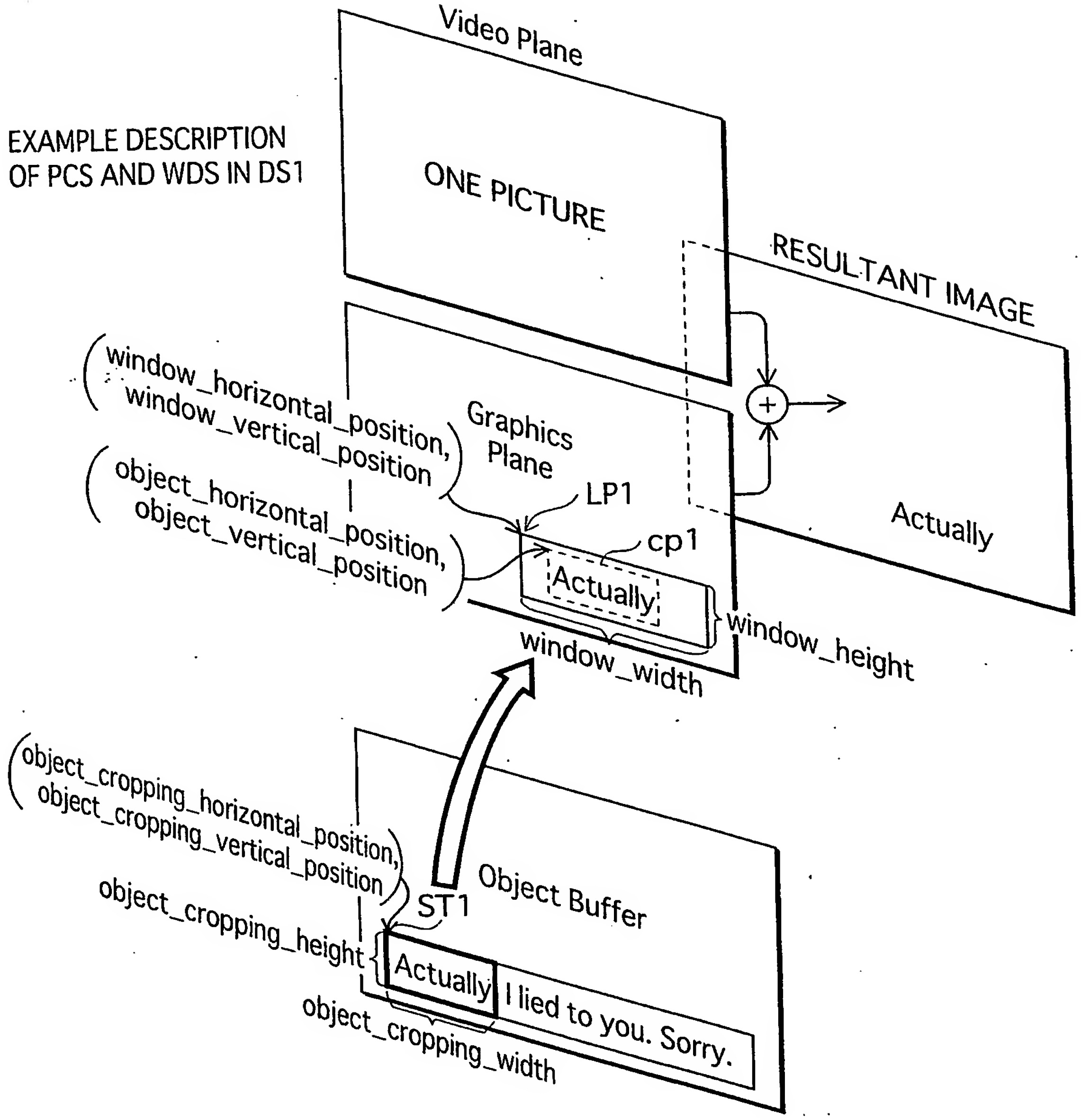

EXAMPLE DESCRIPTION OF PCS AND WDS IN DS1
Video Plane
ONE PICTURE
RESULTANT IMAGE
Actually
window_horizontal_position,
window_vertical_position
Graphics Plane
object_horizontal_position,
object_vertical_position
LP1
cp1
Actually
window_height
window_width
object_cropping_horizontal_position,
object_cropping_vertical_position
ST1
Object Buffer
object_cropping_height
Actually
I lied to you. Sorry.
object_cropping_width

FIG.11

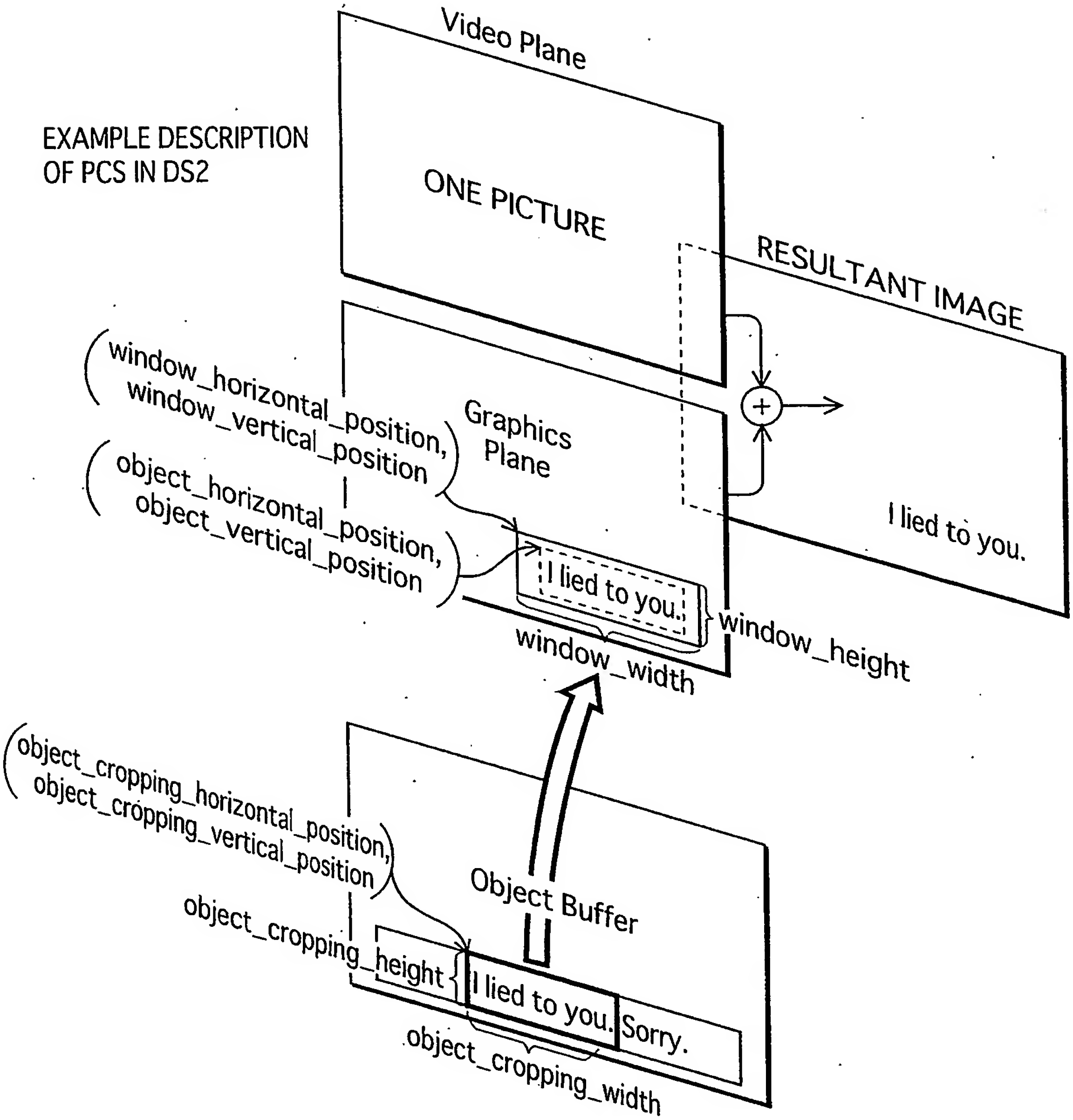

Video Plane
EXAMPLE DESCRIPTION OF PCS IN DS2
ONE PICTURE
RESULTANT IMAGE
(window_horizontal_position, window_vertical_position)
Graphics Plane
(object_horizontal_position, object_vertical_position)
I lied to you.
window_height
window_width
I lied to you.
(object_cropping_horizontal_position, object_cropping_vertical_position)
Object Buffer
object_cropping_height
I lied to you.
Sorry.
object_cropping_width

FIG.12

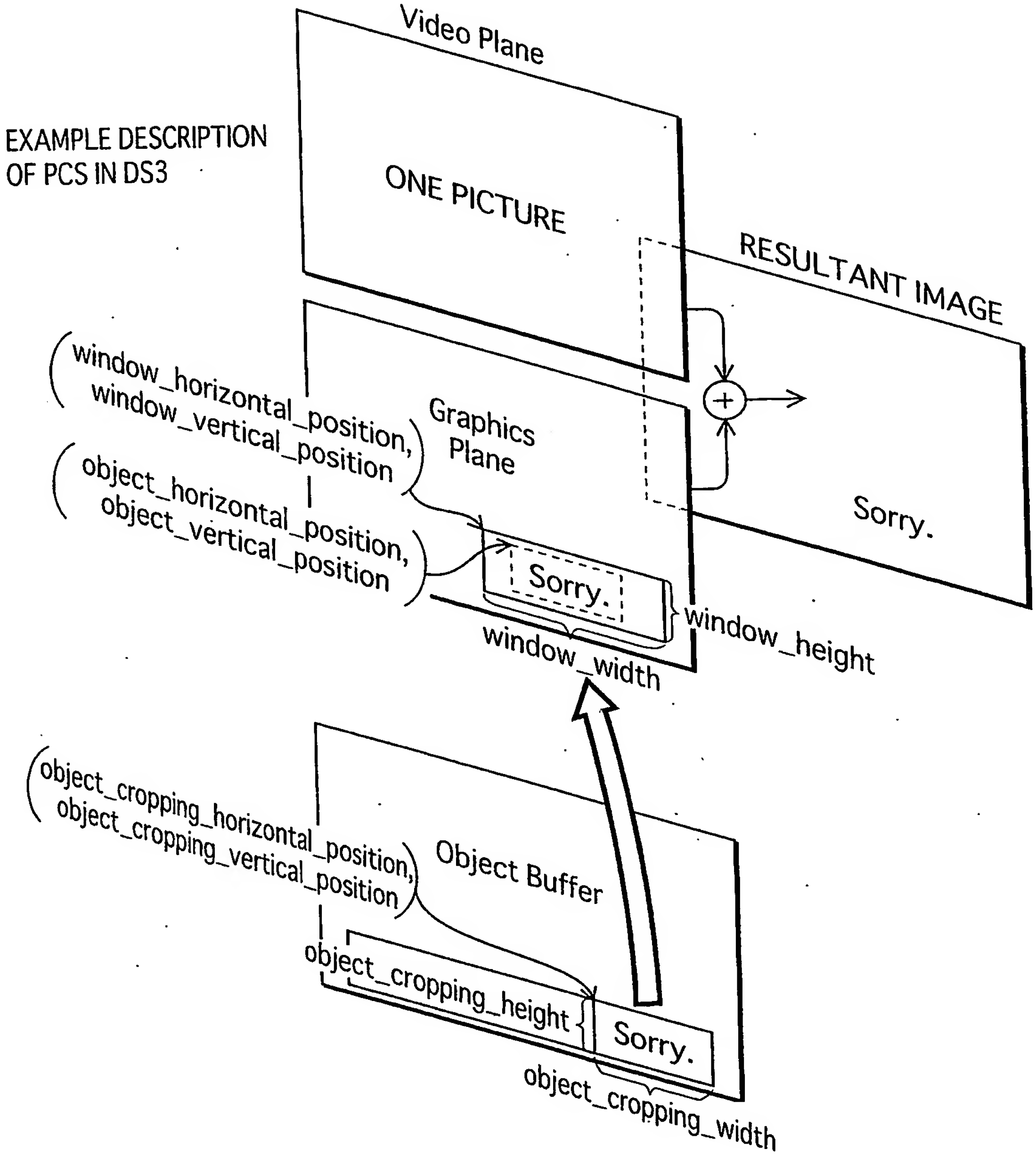

EXAMPLE DESCRIPTION
OF PCS IN DS3
Video Plane
ONE PICTURE
RESULTANT IMAGE
Sorry.
+
window_horizontal_position,
window_vertical_position
Graphics Plane
object_horizontal_position,
object_vertical_position
Sorry.
window_height
window_width
object_cropping_horizontal_position,
object_cropping_vertical_position
Object Buffer
object_cropping_height
Sorry.
object_cropping_width

FIG.13

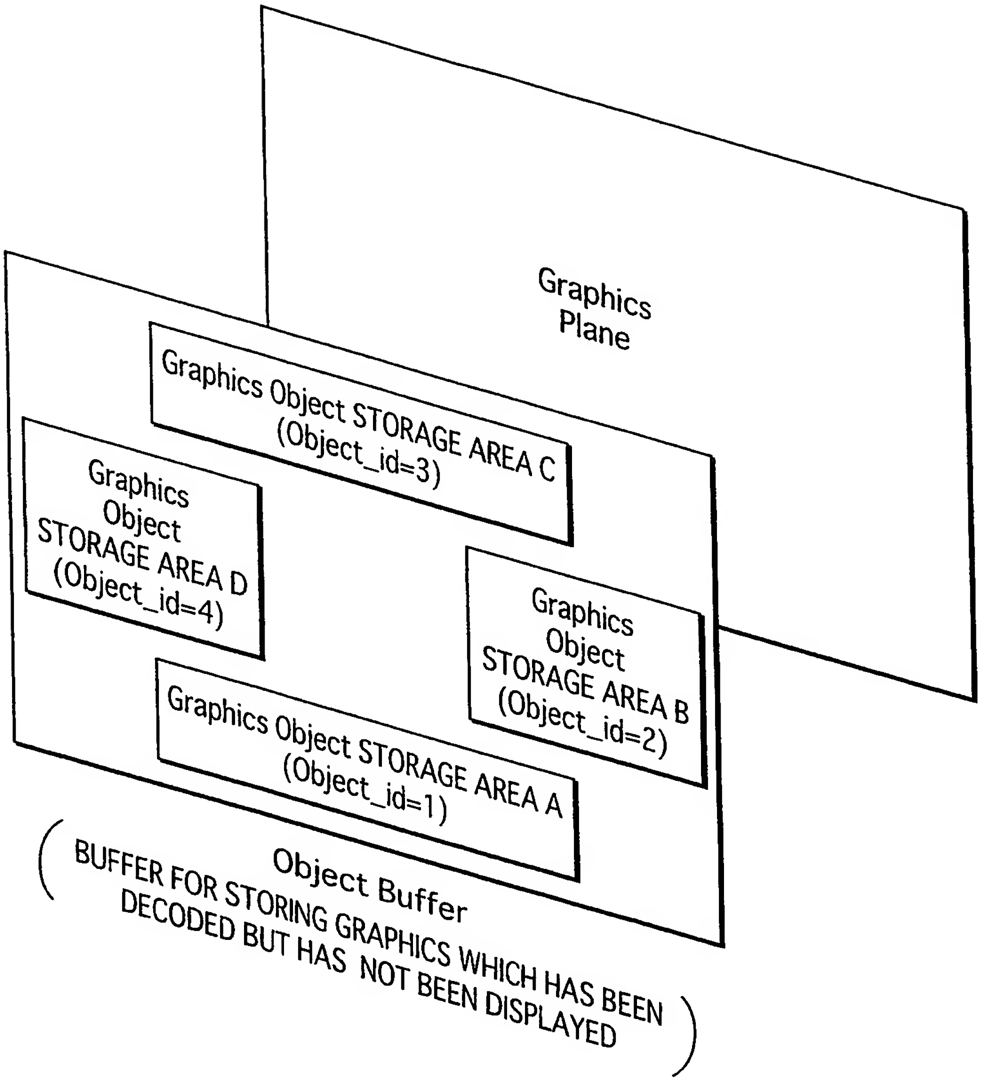

Graphics Plane
Graphics Object STORAGE AREA C (Object_id=3)
Graphics Object STORAGE AREA D (Object_id=4)
Graphics Object STORAGE AREA B (Object_id=2)
Graphics Object STORAGE AREA A (Object_id=1)
Object Buffer
(BUFFER FOR STORING GRAPHICS WHICH HAS BEEN DECODED BUT HAS NOT BEEN DISPLAYED)

FIG.14

```
PTS( DSn[PCS)] )>=DTS( DSn[PCS] )+DECODEDURATION( DSn )
Where:
•  DECODEDURATION( DSn ) is calculated as follows:
decode_duration = 0 ;
decode_duration += PLANEINITIALIZATIONTIME( DSn ) ;
if( DSn. PCS. num_of_objects == 2 )
{
    decode_duration += WAIT( DSn, DSn. PCS. OBJ[0], decode_duration ) ;
    if( DSn. PCS. OBJ[0]. window_id == DSn. PCS. OBJ[1]. window_id )
    {
            decode_duration += WAIT( DSn, DSn. PCS. OBJ[1], decode_duration ) ;
            decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*$10^6$ ) ;
    }
    else
    {
            decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*$10^6$ ) ;
            decode_duration += WAIT( DSn, DSn. PCS. OBJ[1], decode_duration );
            decode_duration+= 90000*( SIZE( DSn. PCS. OBJ[1]. window_id )//256*$10^6$ ) ;
    }
}
else if( DSn. PCS. num_of_objects ==1 )
{
    decode_duration += WAIT( DSn, DSn. PCS. OBJ[0], decode_duration ) ;
    decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*$10^6$ ) ;
}
return decode_duration ;

•  PLANEINITIALIZATIONTIME( DSn ) is calculated as follows:
initialize_duration=0 ;
if( DSn. PCS. composition_state= = EPOCH_START )
{
  initialize_duration = 90000*( 8*video_width*video_height//256*$10^6$) ;
}

else
 {
    for( i=0 ; i < WDS. num_windows ; i++ )
    {
            if( EMPTY(DSn.WDS.WIN[i],DSn ) )
                   initialize_duration += 90000*( SIZE( DSn. WDS. WIN[i] )//256*$10^6$) ;
    }
}
return initialize_duration ;

•  WAIT( DSn, OBJ, current_duration ) is calculated as follows:
wait_duration = 0 ;
if( EXISTS( OBJ. object_id, DSn ) )
{
    object_definition_ready_time = PTS( GET( OBJ. object_id. DSn ) ) ;
    current_time = DTS( DSn. PCS )+current_duration ;
    if( current_time < object_definition_ready_time )
            wait_duration += object_definition_ready_time - current_time ) ;
}
return wait_duration ;
```

FIG.15

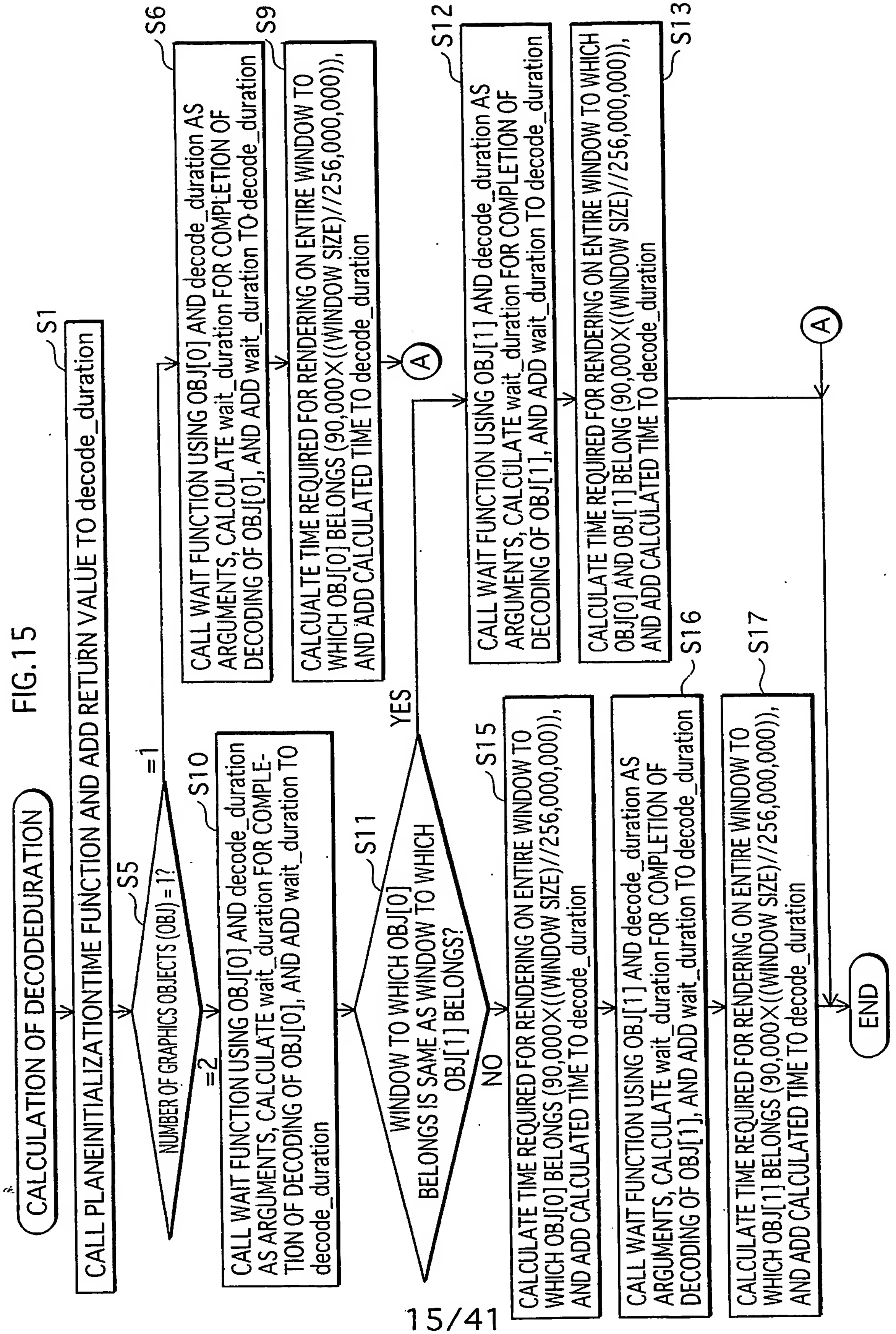

CALCULATION OF DECODEDURATION
S1
CALL PLANEINITIALIZATIONTIME FUNCTION AND ADD RETURN VALUE TO decode_duration
S5
NUMBER OF GRAPHICS OBJECTS (OBJ) = 1?
=1
=2
S6
CALL WAIT FUNCTION USING OBJ[0] AND decode_duration AS ARGUMENTS, CALCULATE wait_duration FOR COMPLETION OF DECODING OF OBJ[0], AND ADD wait_duration TO decode_duration
S9
CALCUALTE TIME REQUIRED FOR RENDERING ON ENTIRE WINDOW TO WHICH OBJ[0] BELONGS (90,000×((WINDOW SIZE)//256,000,000)), AND ADD CALCULATED TIME TO decode_duration
A
S10
CALL WAIT FUNCTION USING OBJ[0] AND decode_duration AS ARGUMENTS, CALCULATE wait_duration FOR COMPLETION OF DECODING OF OBJ[0], AND ADD wait_duration TO decode_duration
S11
WINDOW TO WHICH OBJ[0] BELONGS IS SAME AS WINDOW TO WHICH OBJ[1] BELONGS?
YES
NO
S12
CALL WAIT FUNCTION USING OBJ[1] AND decode_duration AS ARGUMENTS, CALCULATE wait_duration FOR COMPLETION OF DECODING OF OBJ[1], AND ADD wait_duration TO decode_duration
S13
CALCULATE TIME REQUIRED FOR RENDERING ON ENTIRE WINDOW TO WHICH OBJ[0] AND OBJ[1] BELONG (90,000×((WINDOW SIZE)//256,000,000)), AND ADD CALCULATED TIME TO decode_duration
A
S15
CALCULATE TIME REQUIRED FOR RENDERING ON ENTIRE WINDOW TO WHICH OBJ[0] BELONGS (90,000×((WINDOW SIZE)//256,000,000)), AND ADD CALCULATED TIME TO decode_duration
S16
CALL WAIT FUNCTION USING OBJ[1] AND decode_duration AS ARGUMENTS, CALCULATE wait_duration FOR COMPLETION OF DECODING OF OBJ[1], AND ADD wait_duration TO decode_duration
S17
CALCULATE TIME REQUIRED FOR RENDERING ON ENTIRE WINDOW TO WHICH OBJ[1] BELONGS (90,000×((WINDOW SIZE)//256,000,000)), AND ADD CALCULATED TIME TO decode_duration
END

FIG.16A

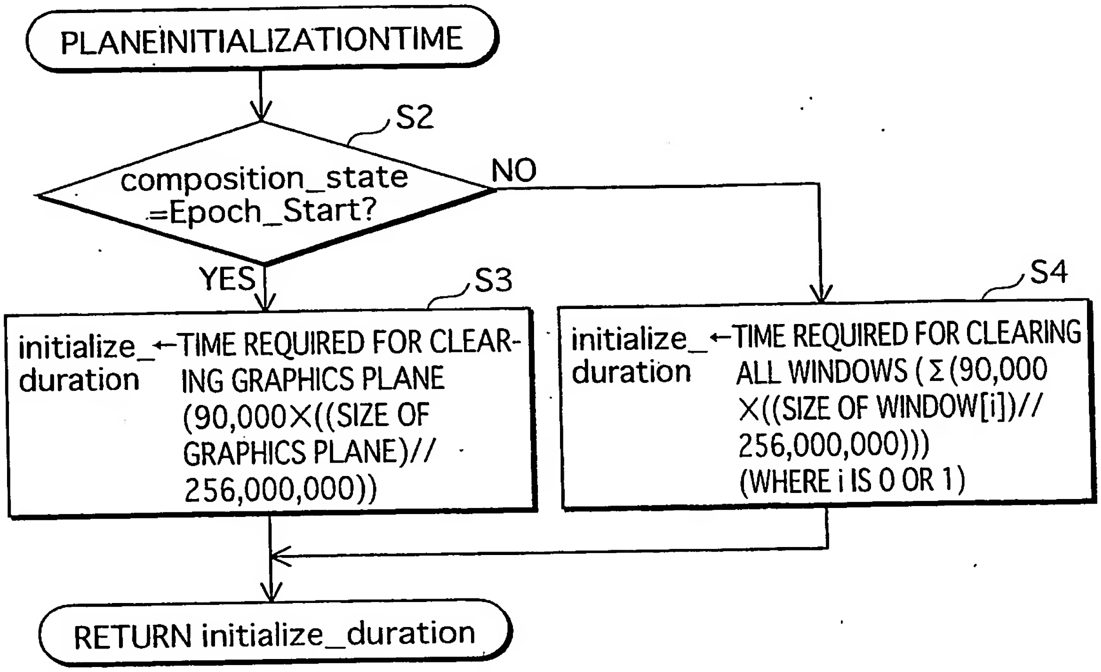

PLANEINITIALIZATIONTIME
S2
composition_state =Epoch_Start?
NO
YES
S3
initialize_ duration ←TIME REQUIRED FOR CLEARING GRAPHICS PLANE (90,000×((SIZE OF GRAPHICS PLANE)// 256,000,000))
S4
initialize_ duration ←TIME REQUIRED FOR CLEARING ALL WINDOWS (Σ(90,000 ×((SIZE OF WINDOW[i])// 256,000,000))) (WHERE i IS 0 OR 1)
RETURN initialize_duration

FIG.16B

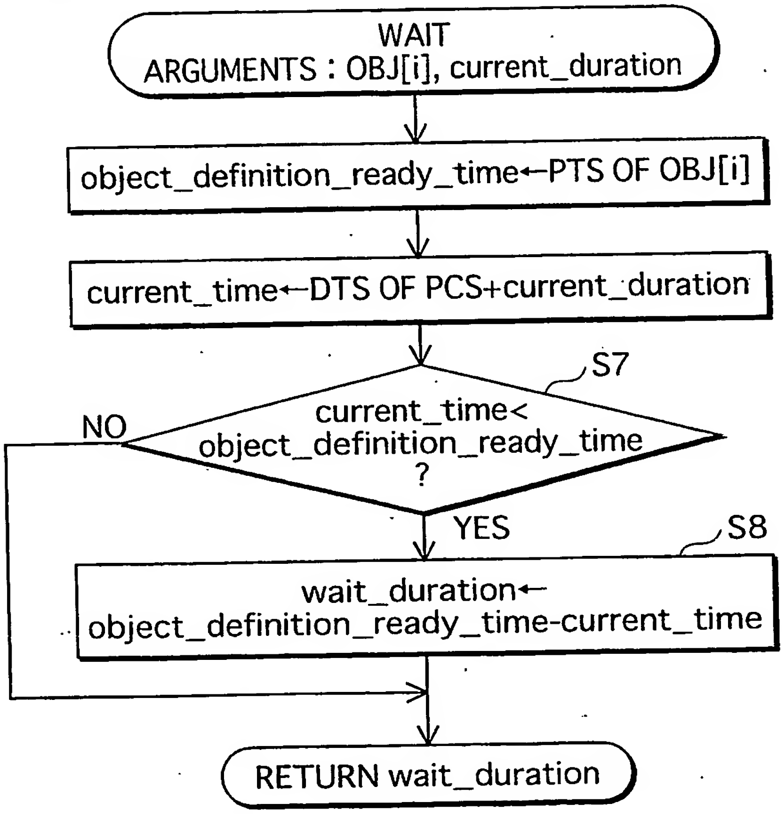

WAIT
ARGUMENTS : OBJ[i], current_duration
object_definition_ready_time←PTS OF OBJ[i]
current_time←DTS OF PCS+current_duration
S7
current_time< object_definition_ready_time ?
NO
YES
S8
wait_duration← object_definition_ready_time-current_time
RETURN wait_duration

FIG.17A

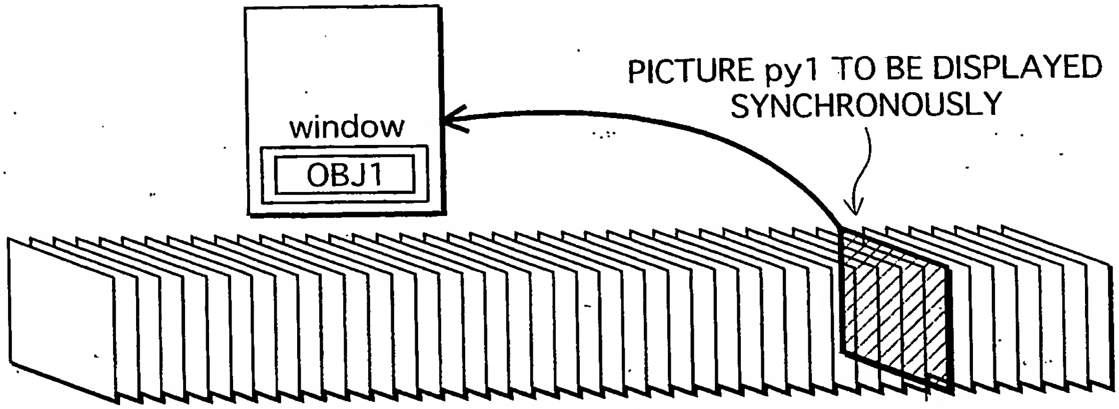


FIG.17B

DECODE_DURATION
=(2)+(3)

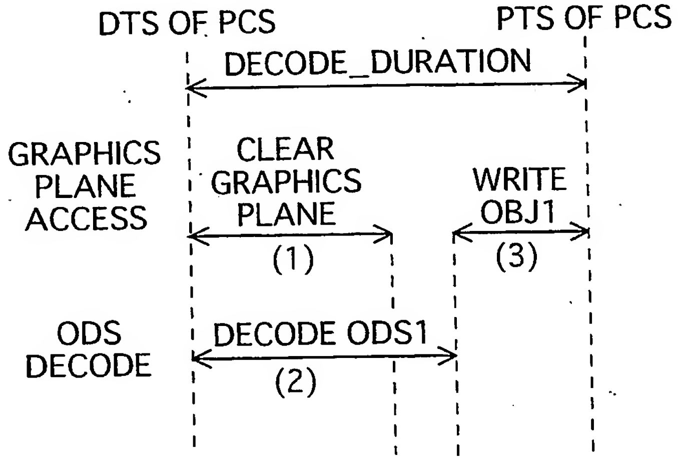


FIG.17C

DECODE_DURATION
=(1)+(3)

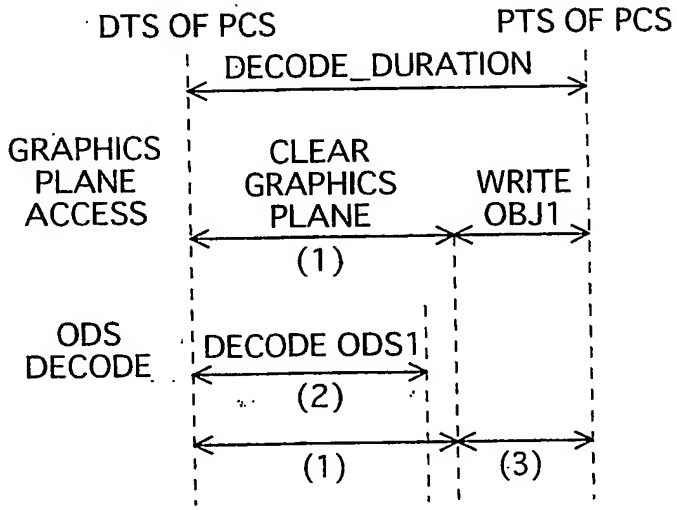


FIG.18A

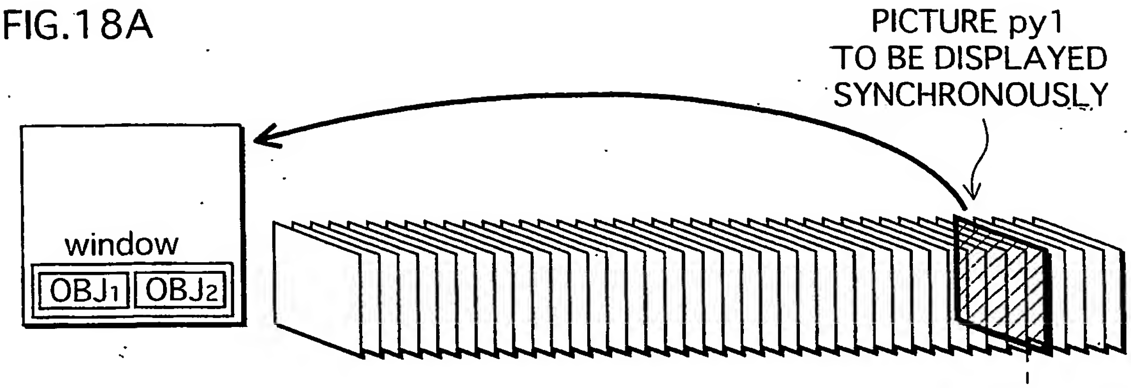


FIG.18B

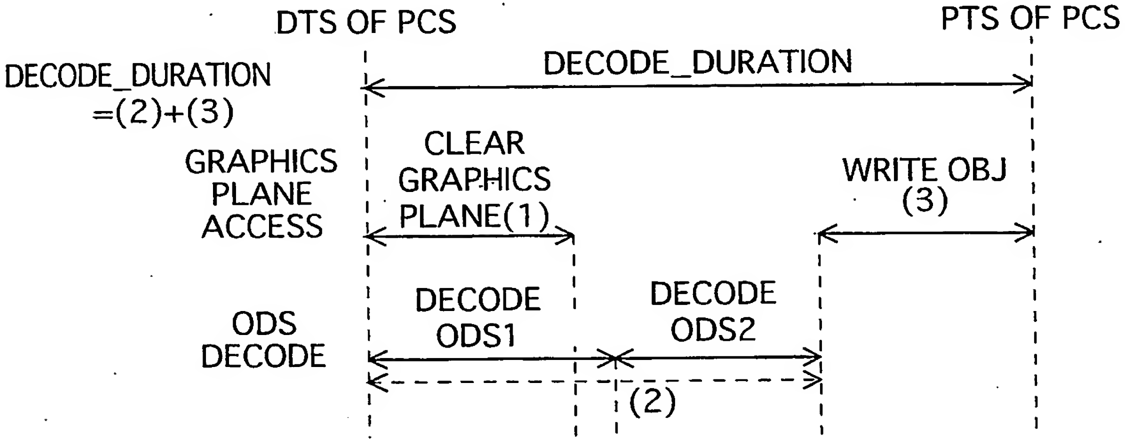


FIG.18C

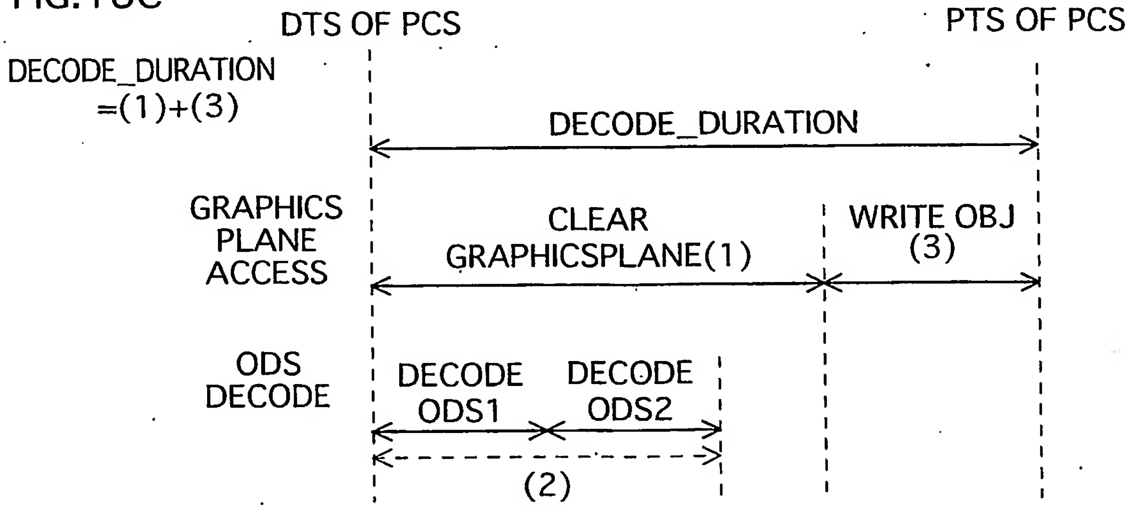


FIG.19A

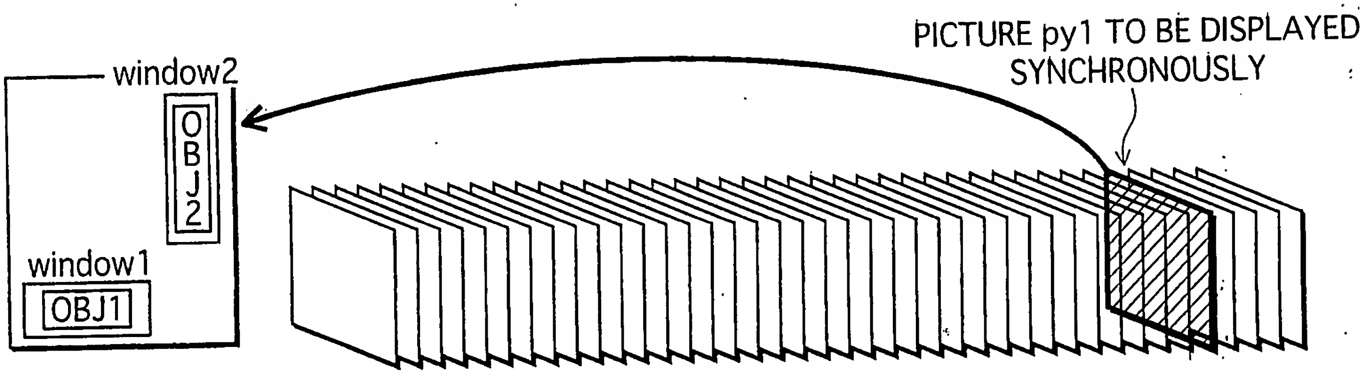


FIG.19B

DECODE_DURATION
=(2)+(32)

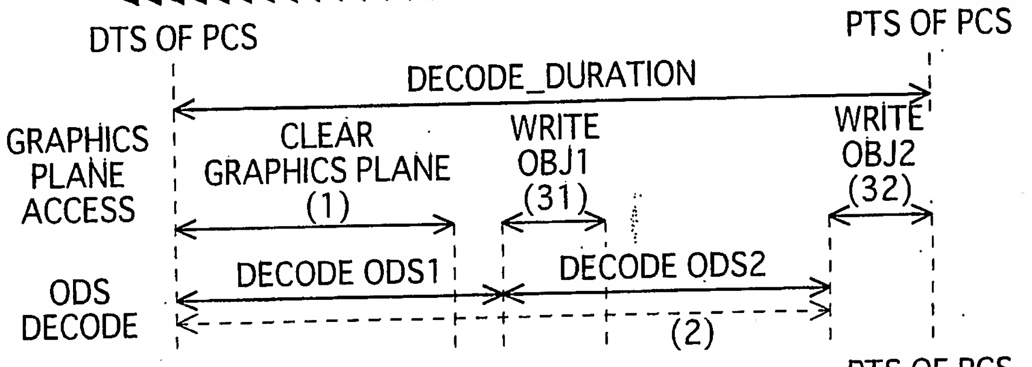


FIG.19C

DECODE_DURATION
=(1)+(31)+(32)

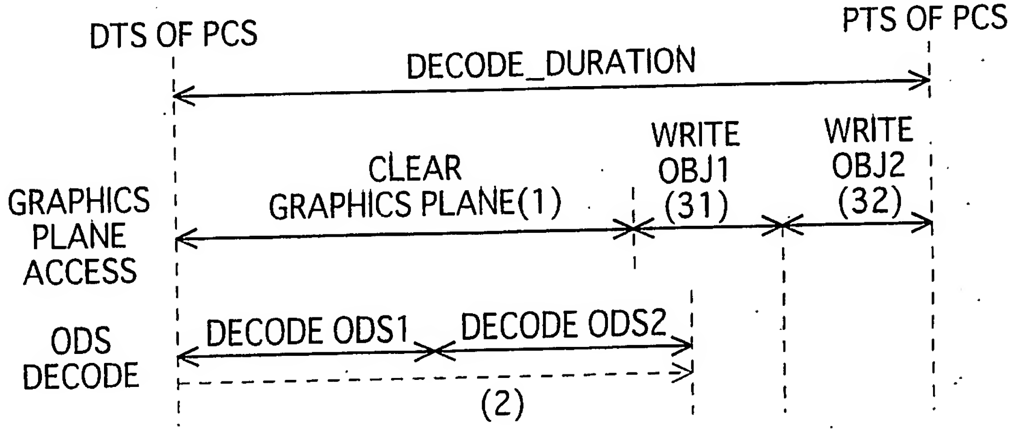


FIG.20

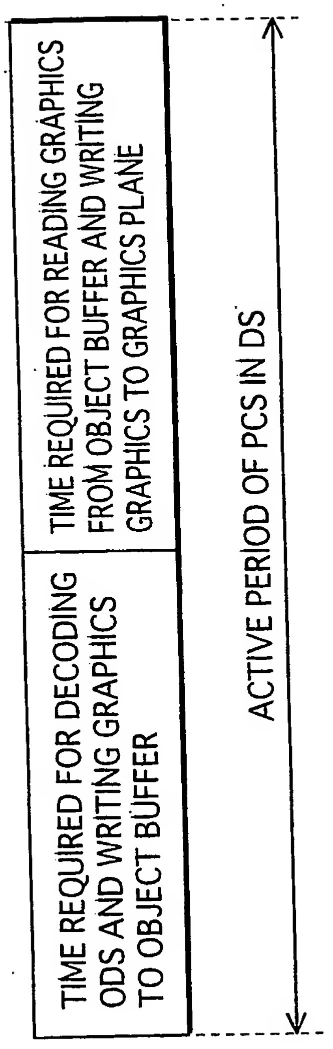

TIME REQUIRED FOR DECODING ODS AND WRITING GRAPHICS TO OBJECT BUFFER
TIME REQUIRED FOR READING GRAPHICS FROM OBJECT BUFFER AND WRITING GRAPHICS TO GRAPHICS PLANE
ACTIVE PERIOD OF PCS IN DS

FIG.21

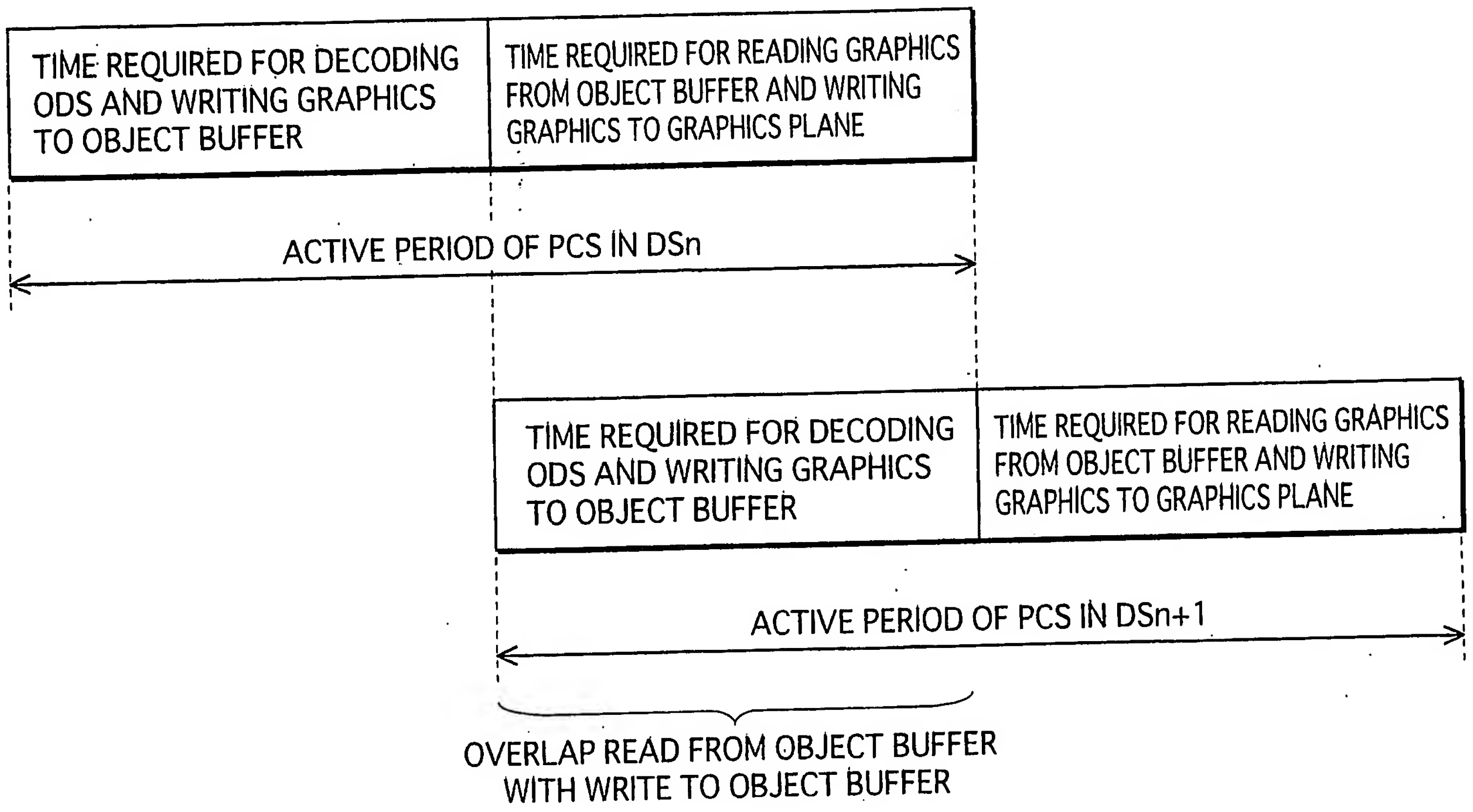

TIME REQUIRED FOR DECODING ODS AND WRITING GRAPHICS TO OBJECT BUFFER
TIME REQUIRED FOR READING GRAPHICS FROM OBJECT BUFFER AND WRITING GRAPHICS TO GRAPHICS PLANE
ACTIVE PERIOD OF PCS IN DSn
TIME REQUIRED FOR DECODING ODS AND WRITING GRAPHICS TO OBJECT BUFFER
TIME REQUIRED FOR READING GRAPHICS FROM OBJECT BUFFER AND WRITING GRAPHICS TO GRAPHICS PLANE
ACTIVE PERIOD OF PCS IN DSn+1
OVERLAP READ FROM OBJECT BUFFER WITH WRITE TO OBJECT BUFFER

FIG.22

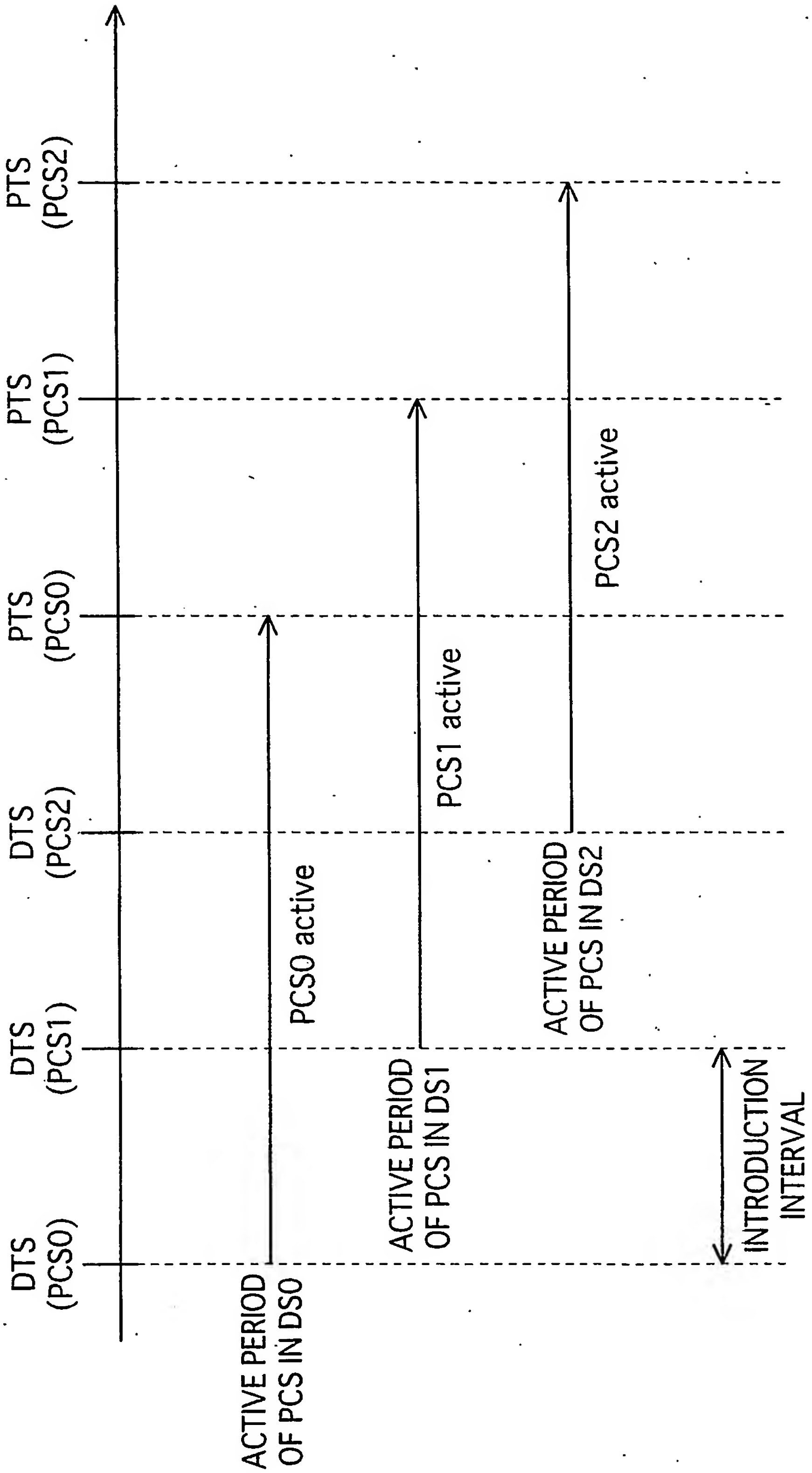

DTS (PCS0)
DTS (PCS1)
DTS (PCS2)
PTS (PCS0)
PTS (PCS1)
PTS (PCS2)
ACTIVE PERIOD OF PCS IN DS0
PCS0 active
ACTIVE PERIOD OF PCS IN DS1
PCS1 active
ACTIVE PERIOD OF PCS IN DS2
PCS2 active
INTRODUCTION INTERVAL

FIG.23

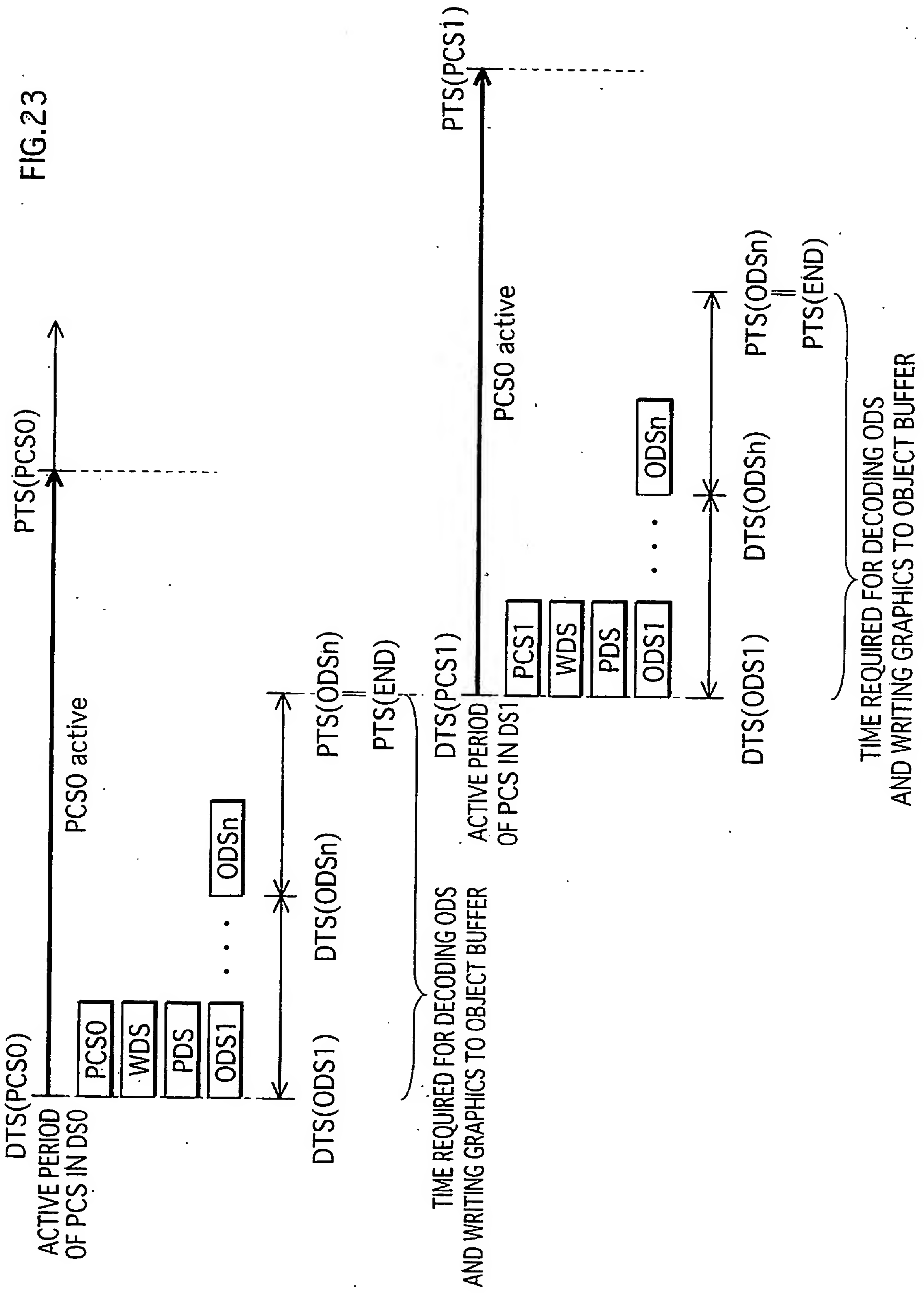

DTS(PCS0)
ACTIVE PERIOD
OF PCS IN DS0
PCS0
WDS
PDS
ODS1
ODSn
DTS(ODS1)
DTS(ODSn)
PTS(ODSn)
PTS(END)
PCS0 active
PTS(PCS0)
TIME REQUIRED FOR DECODING ODS
AND WRITING GRAPHICS TO OBJECT BUFFER
DTS(PCS1)
ACTIVE PERIOD
OF PCS IN DS1
PCS1
WDS
PDS
ODS1
ODSn
DTS(ODS1)
DTS(ODSn)
PTS(ODSn)
PTS(END)
PCS0 active
PTS(PCS1)
TIME REQUIRED FOR DECODING ODS
AND WRITING GRAPHICS TO OBJECT BUFFER

FIG.24

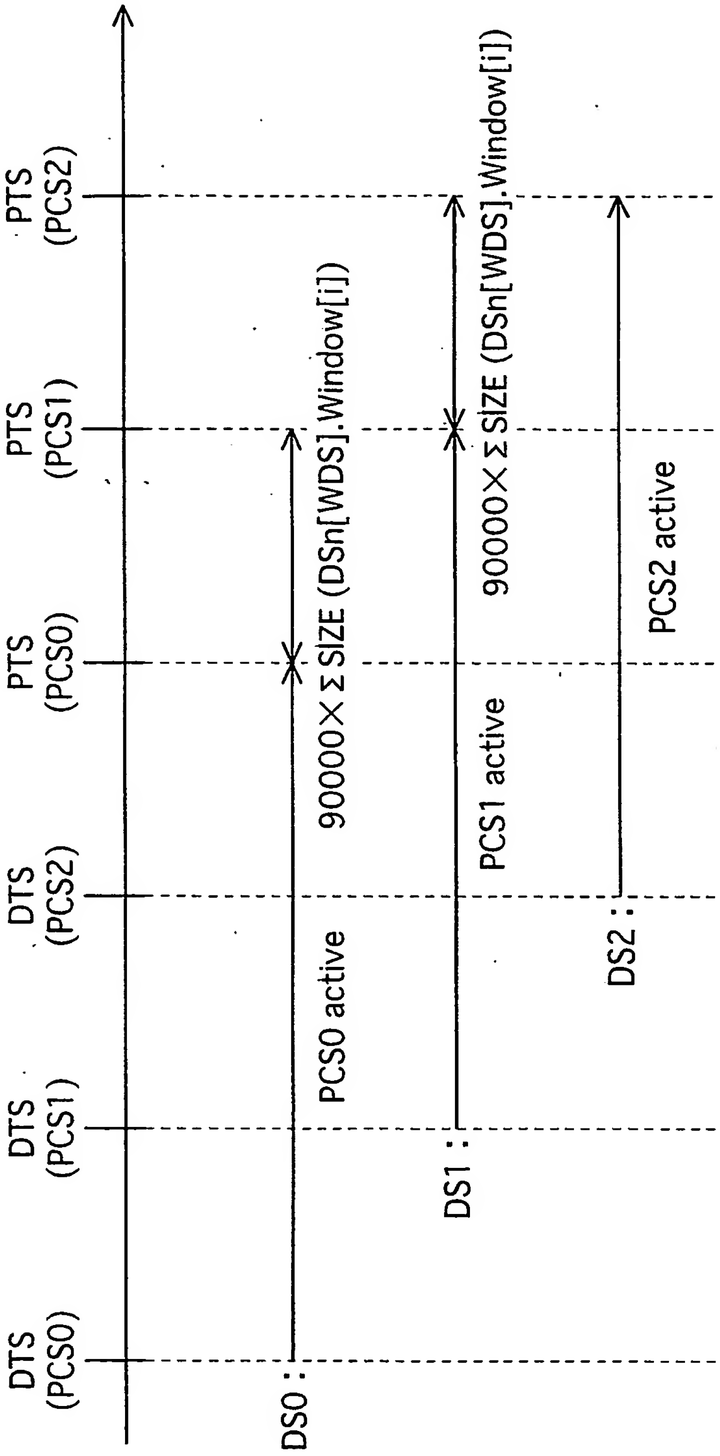

DTS (PCS0)
DTS (PCS1)
DTS (PCS2)
PTS (PCS0)
PTS (PCS1)
PTS (PCS2)
DS0 :
PCS0 active
90000× Σ SIZE (DSn[WDS].Window[i])
DS1 :
PCS1 active
90000× Σ SIZE (DSn[WDS].Window[i])
DS2 :
PCS2 active

FIG.25A PIPELINE

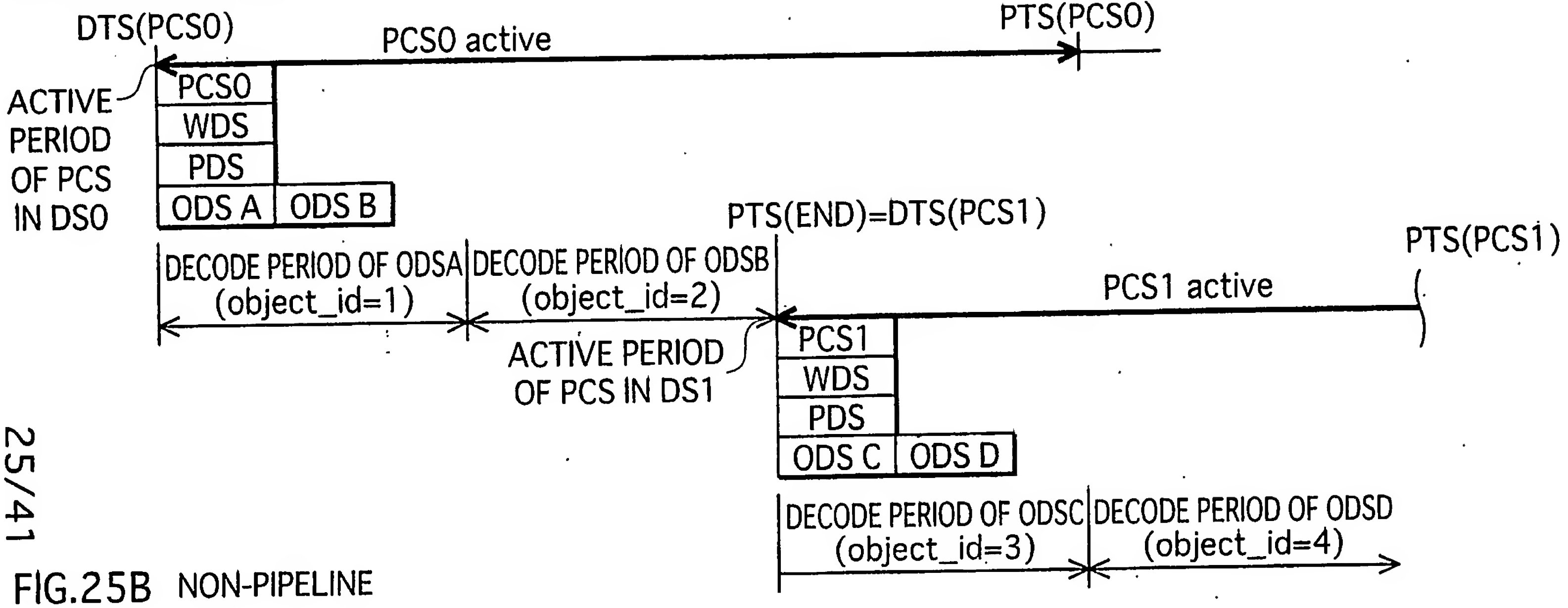

DTS(PCS0)
PCS0 active
PTS(PCS0)
ACTIVE PERIOD OF PCS IN DS0
PCS0
WDS
PDS
ODS A
ODS B
DECODE PERIOD OF ODSA (object_id=1)
DECODE PERIOD OF ODSB (object_id=2)
PTS(END)=DTS(PCS1)
PTS(PCS1)
PCS1 active
ACTIVE PERIOD OF PCS IN DS1
PCS1
WDS
PDS
ODS C
ODS D
DECODE PERIOD OF ODSC (object_id=3)
DECODE PERIOD OF ODSD (object_id=4)

FIG.25B NON-PIPELINE

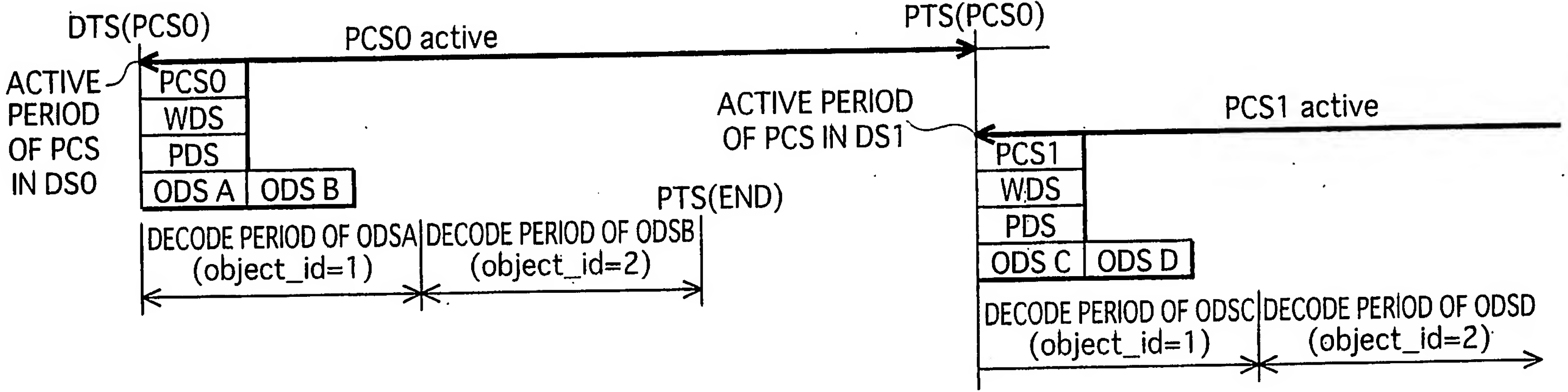

DTS(PCS0)
PCS0 active
PTS(PCS0)
ACTIVE PERIOD OF PCS IN DS0
PCS0
WDS
PDS
ODS A
ODS B
PTS(END)
DECODE PERIOD OF ODSA (object_id=1)
DECODE PERIOD OF ODSB (object_id=2)
ACTIVE PERIOD OF PCS IN DS1
PCS1 active
PCS1
WDS
PDS
ODS C
ODS D
DECODE PERIOD OF ODSC (object_id=1)
DECODE PERIOD OF ODSD (object_id=2)

FIG.26

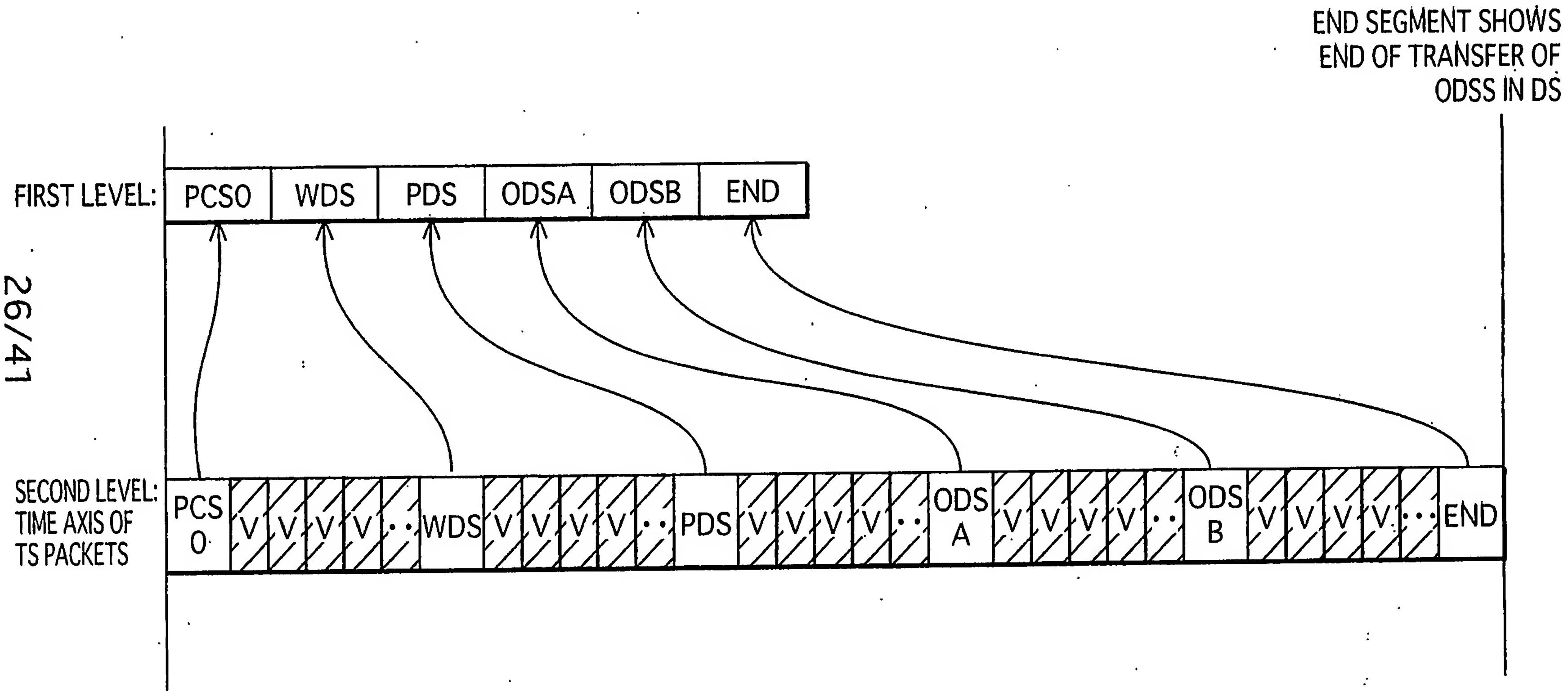

END SEGMENT SHOWS
END OF TRANSFER OF
ODSS IN DS
FIRST LEVEL:
PCS0
WDS
PDS
ODSA
ODSB
END
SECOND LEVEL:
TIME AXIS OF
TS PACKETS
PCS
0
V
V
V
V
WDS
PDS
ODS
A
ODS
B
END

FIG.27A SCREEN COMPOSITION

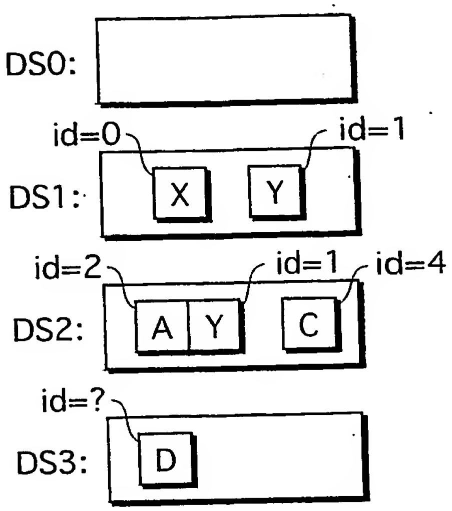


FIG.27B ACTIVE PERIOD OVERLAPPING AND ODS TRANSFER

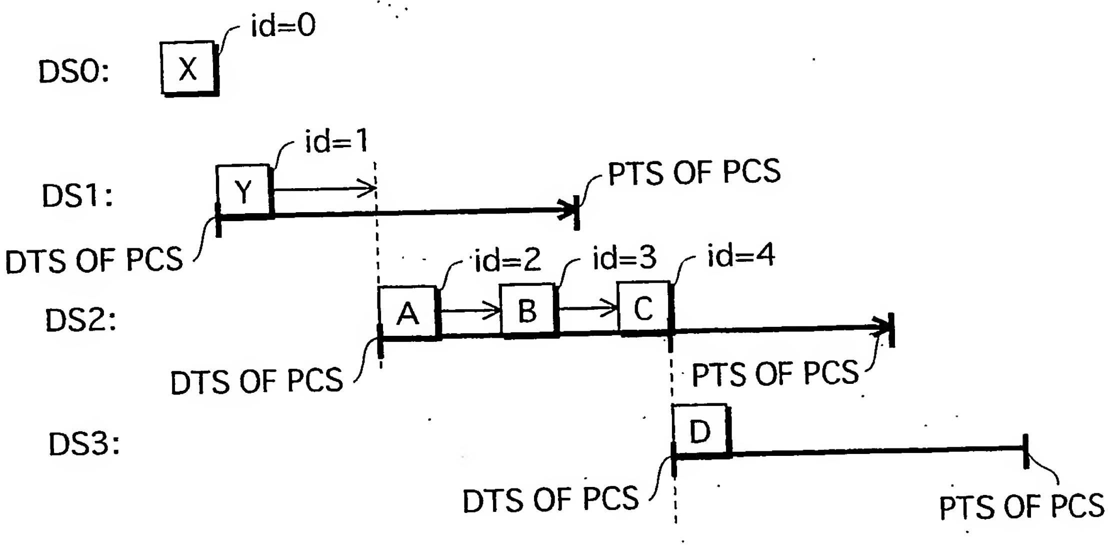


FIG.27C ARRANGEMENT IN OBJECT BUFFER

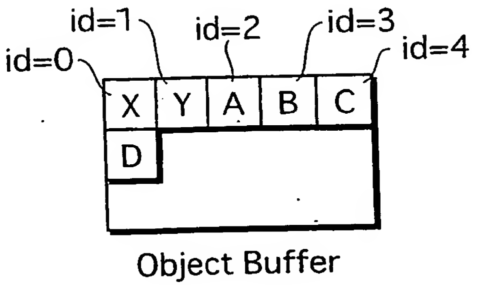


FIG.28

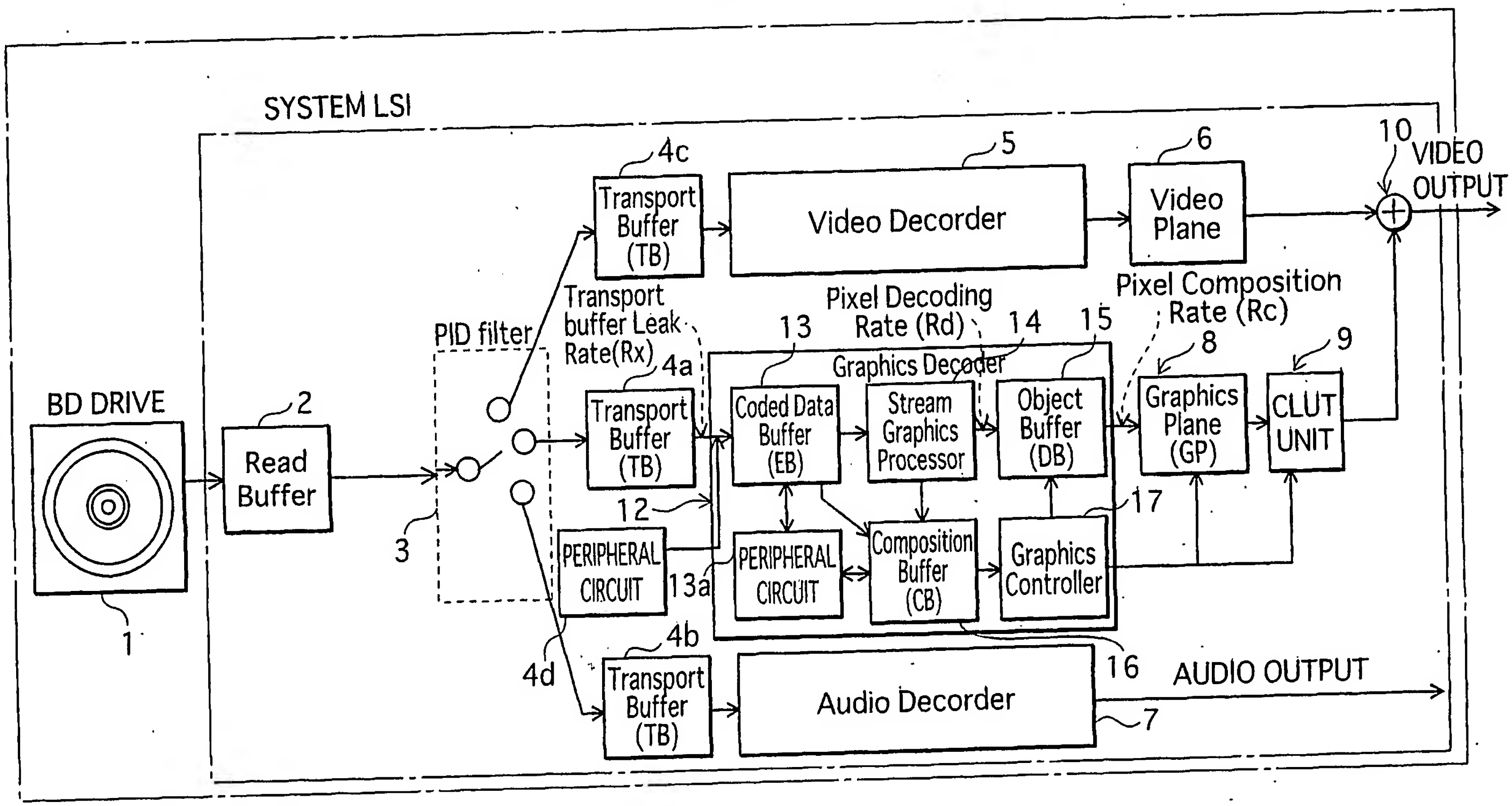

SYSTEM LSI
BD DRIVE
1
Read Buffer
2
3
PID filter
Transport Buffer (TB)
4c
Transport buffer Leak Rate(Rx)
Transport Buffer (TB)
4a
12
PERIPHERAL CIRCUIT
13a
4d
Transport Buffer (TB)
4b
Video Decorder
5
Video Plane
6
Pixel Decoding Rate (Rd)
13
14
15
Graphics Decoder
Coded Data Buffer (EB)
Stream Graphics Processor
Object Buffer (DB)
PERIPHERAL CIRCUIT
Composition Buffer (CB)
Graphics Controller
17
16
Pixel Composition Rate (Rc)
8
Graphics Plane (GP)
9
CLUT UNIT
10
VIDEO OUTPUT
Audio Decorder
7
AUDIO OUTPUT

FIG.29

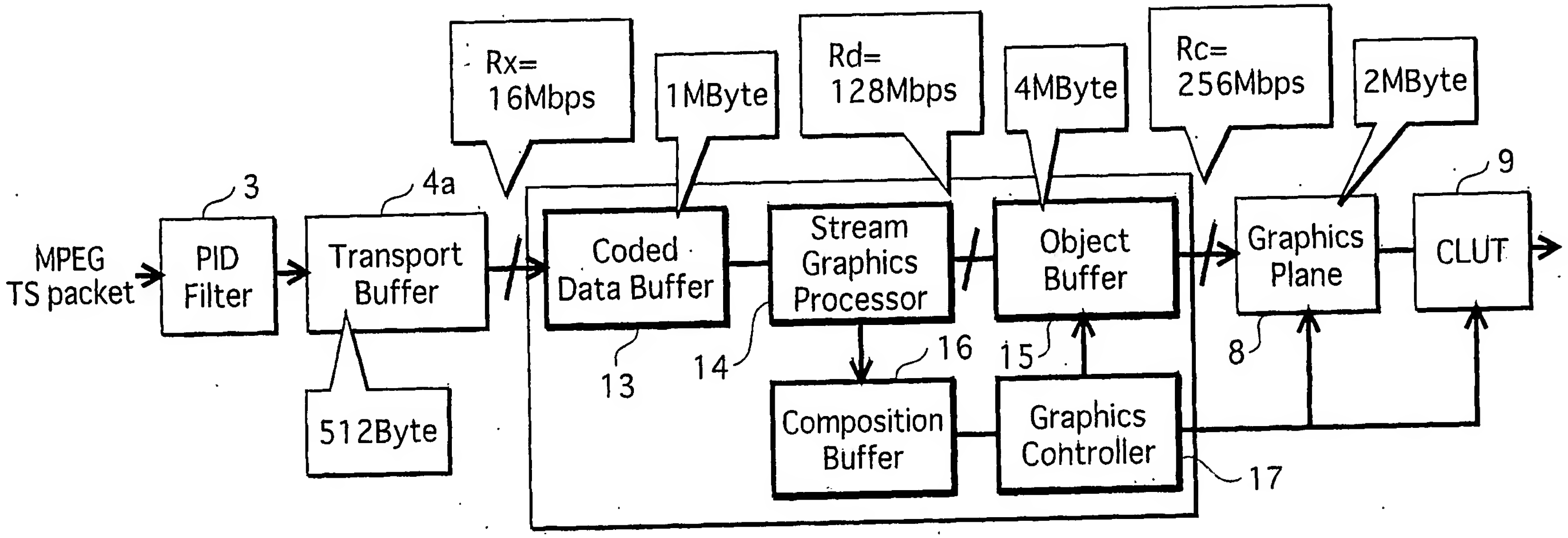

MPEG
TS packet
3
PID Filter
4a
Transport Buffer
512Byte
Rx= 16Mbps
1MByte
Coded Data Buffer
13
14
Stream Graphics Processor
Rd= 128Mbps
16
Composition Buffer
4MByte
Object Buffer
15
Graphics Controller
17
Rc= 256Mbps
2MByte
Graphics Plane
8
9
CLUT

FIG.30

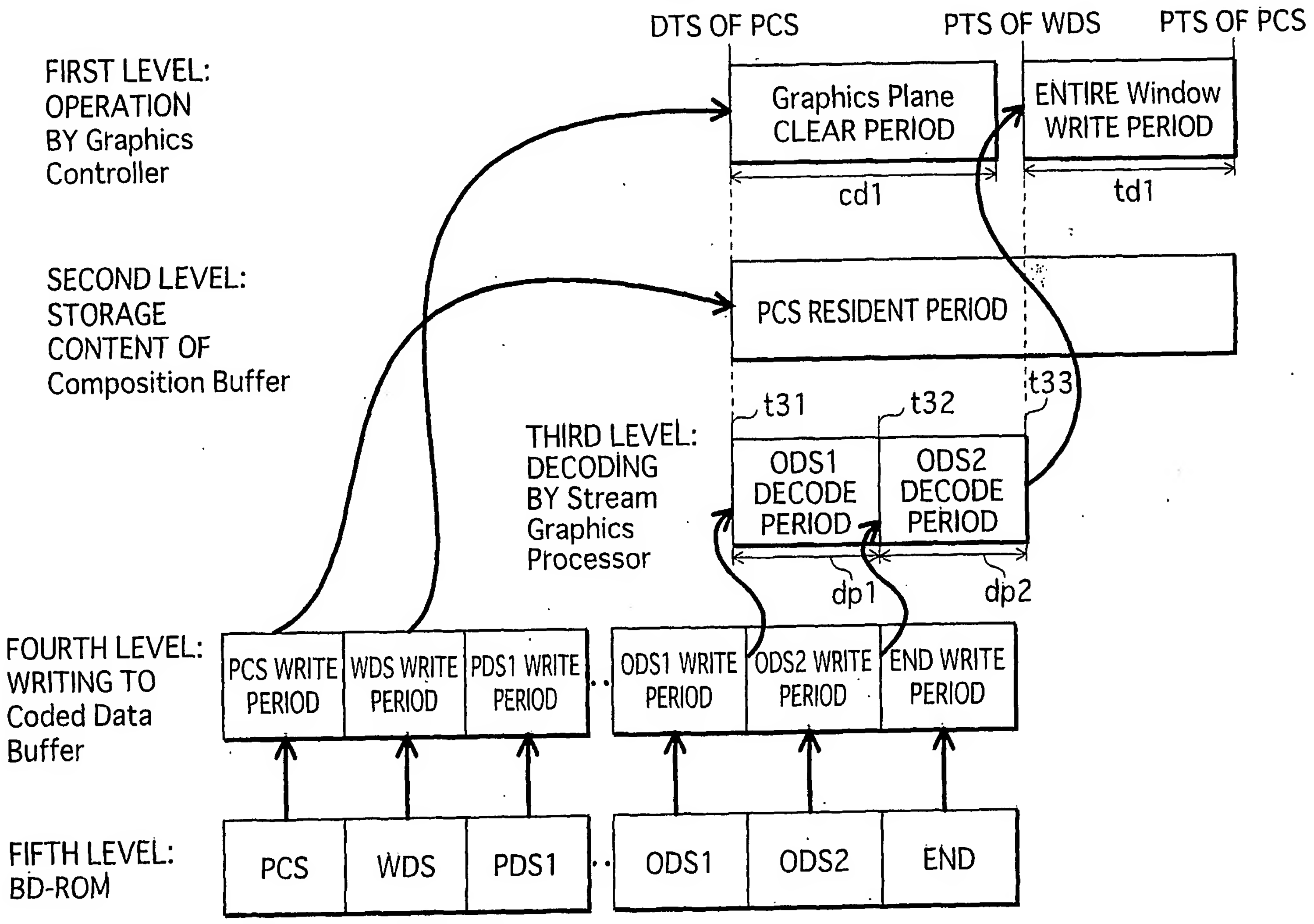

DTS OF PCS
PTS OF WDS
PTS OF PCS
FIRST LEVEL: OPERATION BY Graphics Controller
Graphics Plane CLEAR PERIOD
ENTIRE Window WRITE PERIOD
cd1
td1
SECOND LEVEL: STORAGE CONTENT OF Composition Buffer
PCS RESIDENT PERIOD
t31
t32
t33
THIRD LEVEL: DECODING BY Stream Graphics Processor
ODS1 DECODE PERIOD
ODS2 DECODE PERIOD
dp1
dp2
FOURTH LEVEL: WRITING TO Coded Data Buffer
PCS WRITE PERIOD
WDS WRITE PERIOD
PDS1 WRITE PERIOD
ODS1 WRITE PERIOD
ODS2 WRITE PERIOD
END WRITE PERIOD
FIFTH LEVEL: BD-ROM
PCS
WDS
PDS1
ODS1
ODS2
END

FIG.31

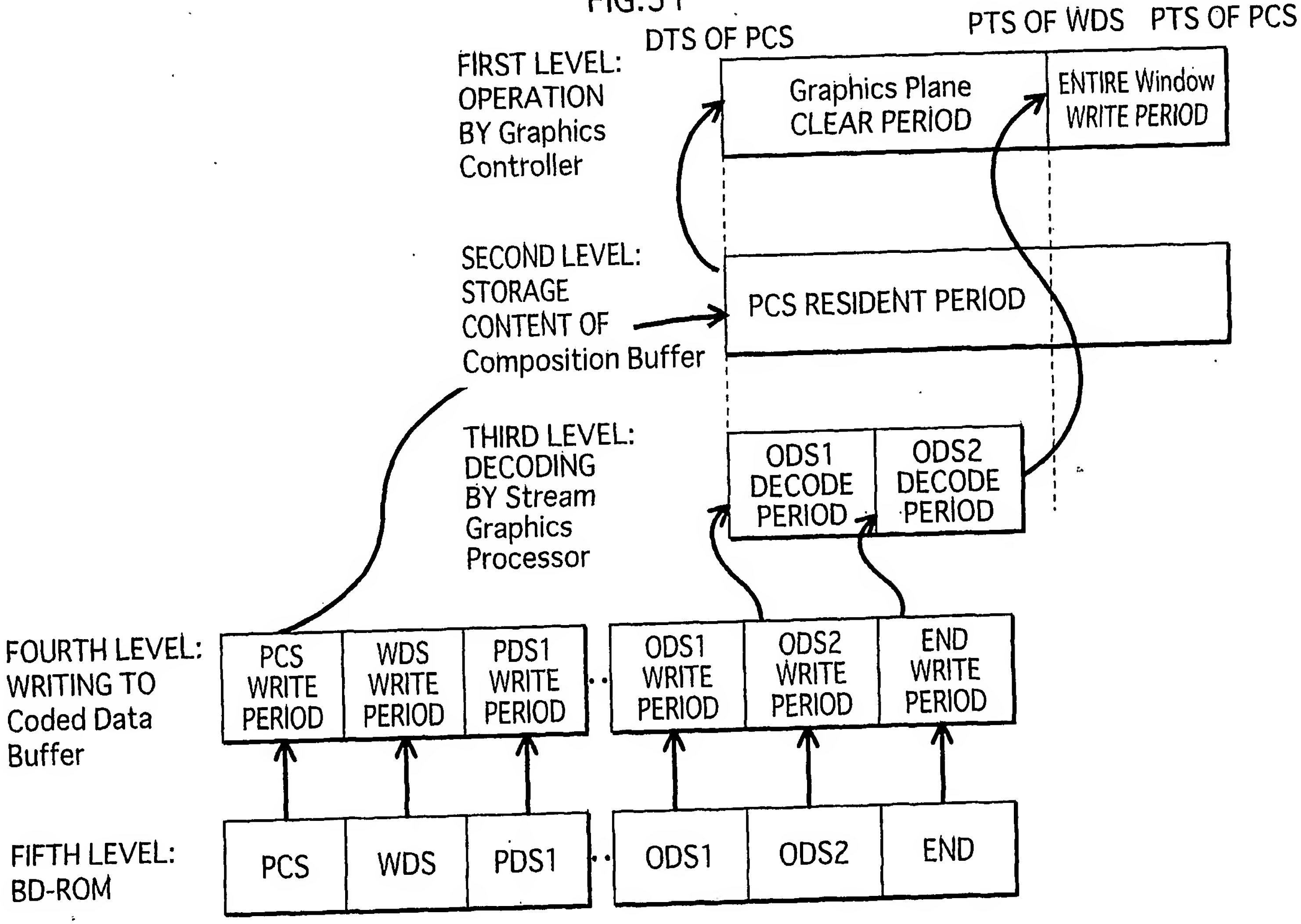

DTS OF PCS
PTS OF WDS
PTS OF PCS
FIRST LEVEL: OPERATION BY Graphics Controller
Graphics Plane CLEAR PERIOD
ENTIRE Window WRITE PERIOD
SECOND LEVEL: STORAGE CONTENT OF Composition Buffer
PCS RESIDENT PERIOD
THIRD LEVEL: DECODING BY Stream Graphics Processor
ODS1 DECODE PERIOD
ODS2 DECODE PERIOD
FOURTH LEVEL: WRITING TO Coded Data Buffer
PCS WRITE PERIOD
WDS WRITE PERIOD
PDS1 WRITE PERIOD
ODS1 WRITE PERIOD
ODS2 WRITE PERIOD
END WRITE PERIOD
FIFTH LEVEL: BD-ROM
PCS
WDS
PDS1
ODS1
ODS2
END

FIG.32

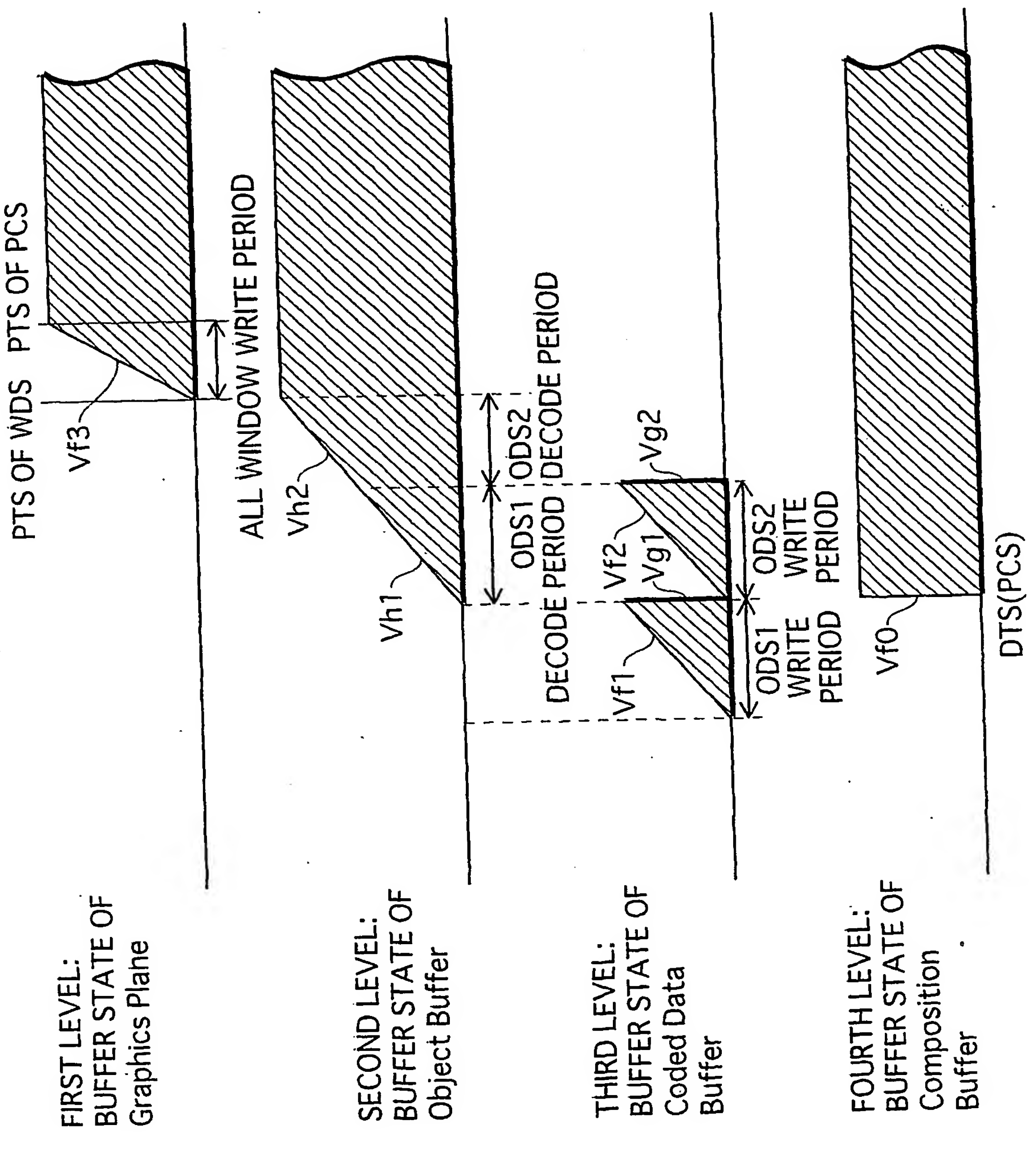

FIRST LEVEL: BUFFER STATE OF Graphics Plane
SECOND LEVEL: BUFFER STATE OF Object Buffer
THIRD LEVEL: BUFFER STATE OF Coded Data Buffer
FOURTH LEVEL: BUFFER STATE OF Composition Buffer
PTS OF WDS
PTS OF PCS
Vf3
ALL WINDOW WRITE PERIOD
Vh2
Vh1
ODS1 DECODE PERIOD
ODS2 DECODE PERIOD
Vf1
Vf2
Vg1
Vg2
ODS1 WRITE PERIOD
ODS2 WRITE PERIOD
Vf0
DTS(PCS)

FIG.33

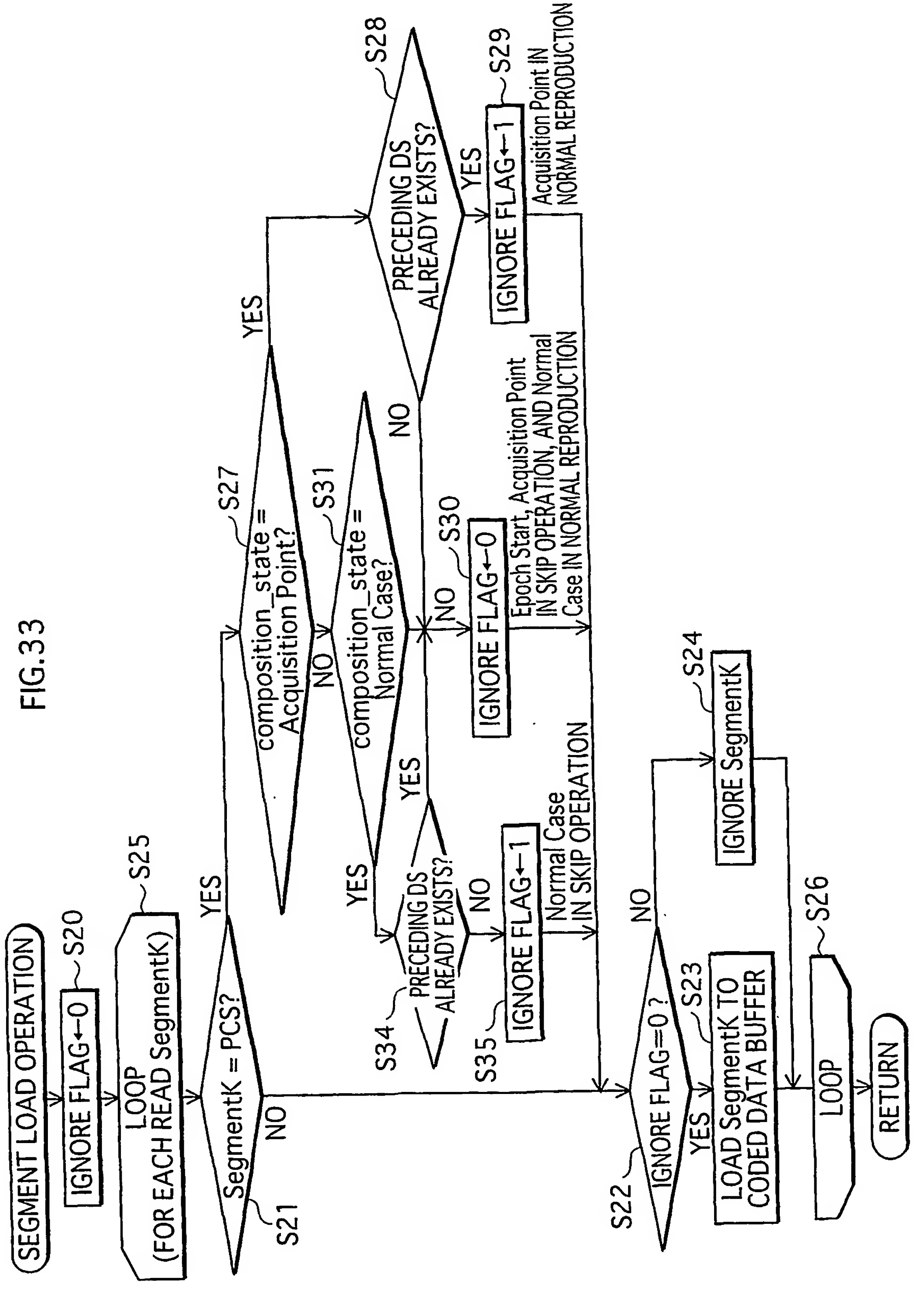

SEGMENT LOAD OPERATION
IGNORE FLAG←0
S20
LOOP (FOR EACH READ SegmentK)
S25
SegmentK = PCS?
S21
YES
NO
composition_state = Acquisition Point?
S27
YES
NO
PRECEDING DS ALREADY EXISTS?
S28
YES
NO
IGNORE FLAG←1
S29
Acquisition Point IN NORMAL REPRODUCTION
composition_state = Normal Case?
S31
YES
NO
IGNORE FLAG←0
S30
Epoch Start, Acquisition Point IN SKIP OPERATION, AND Normal Case IN NORMAL REPRODUCTION
PRECEDING DS ALREADY EXISTS?
S34
YES
NO
IGNORE FLAG←1
S35
Normal Case IN SKIP OPERATION
IGNORE FLAG=0 ?
S22
YES
NO
LOAD SegmentK TO CODED DATA BUFFER
S23
IGNORE SegmentK
S24
LOOP
S26
RETURN

FIG.34

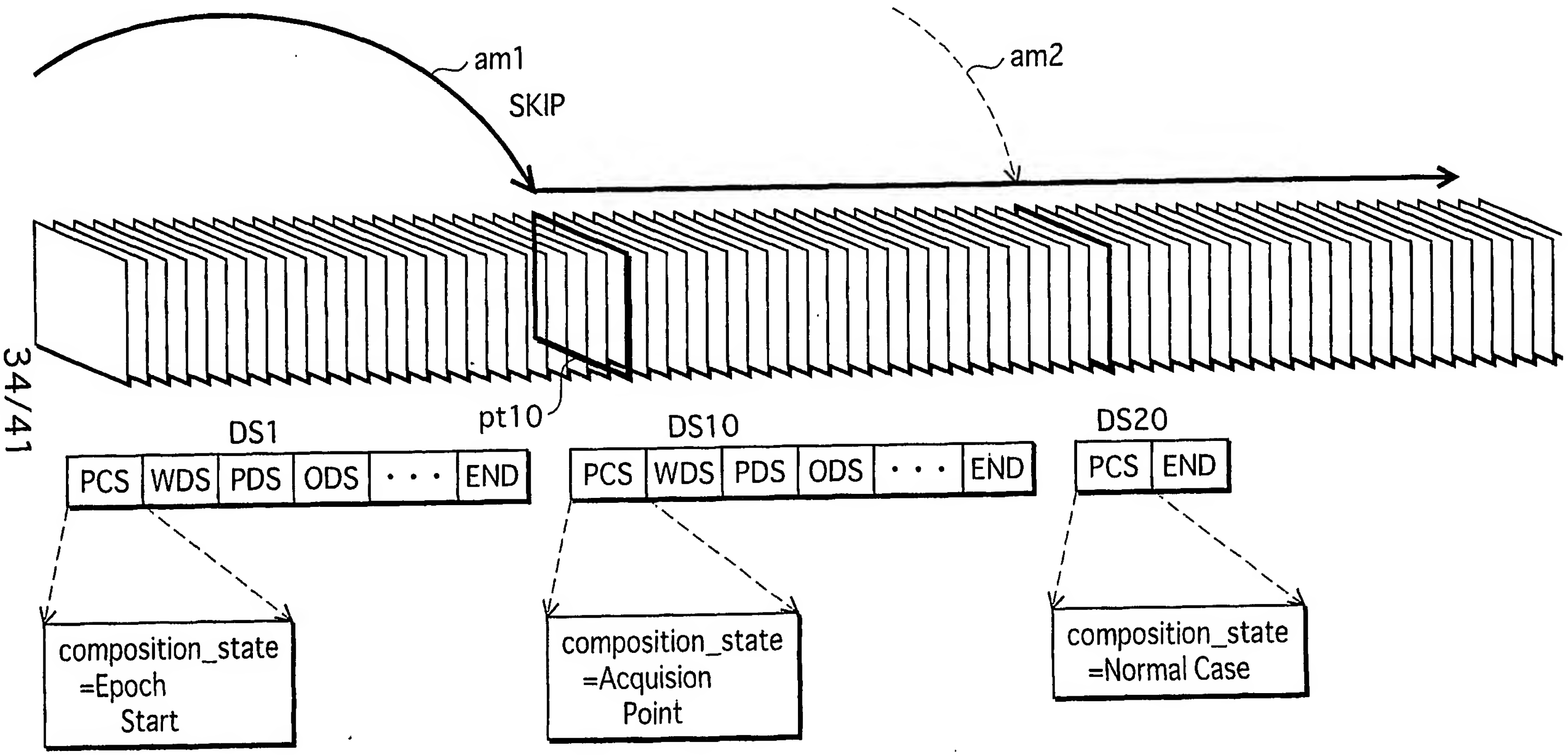

am1
SKIP
am2
pt10
DS1
PCS
WDS
PDS
ODS
· · ·
END
DS10
PCS
WDS
PDS
ODS
· · ·
END
DS20
PCS
END
composition_state
=Epoch
Start
composition_state
=Acquision
Point
composition_state
=Normal Case

FIG.35

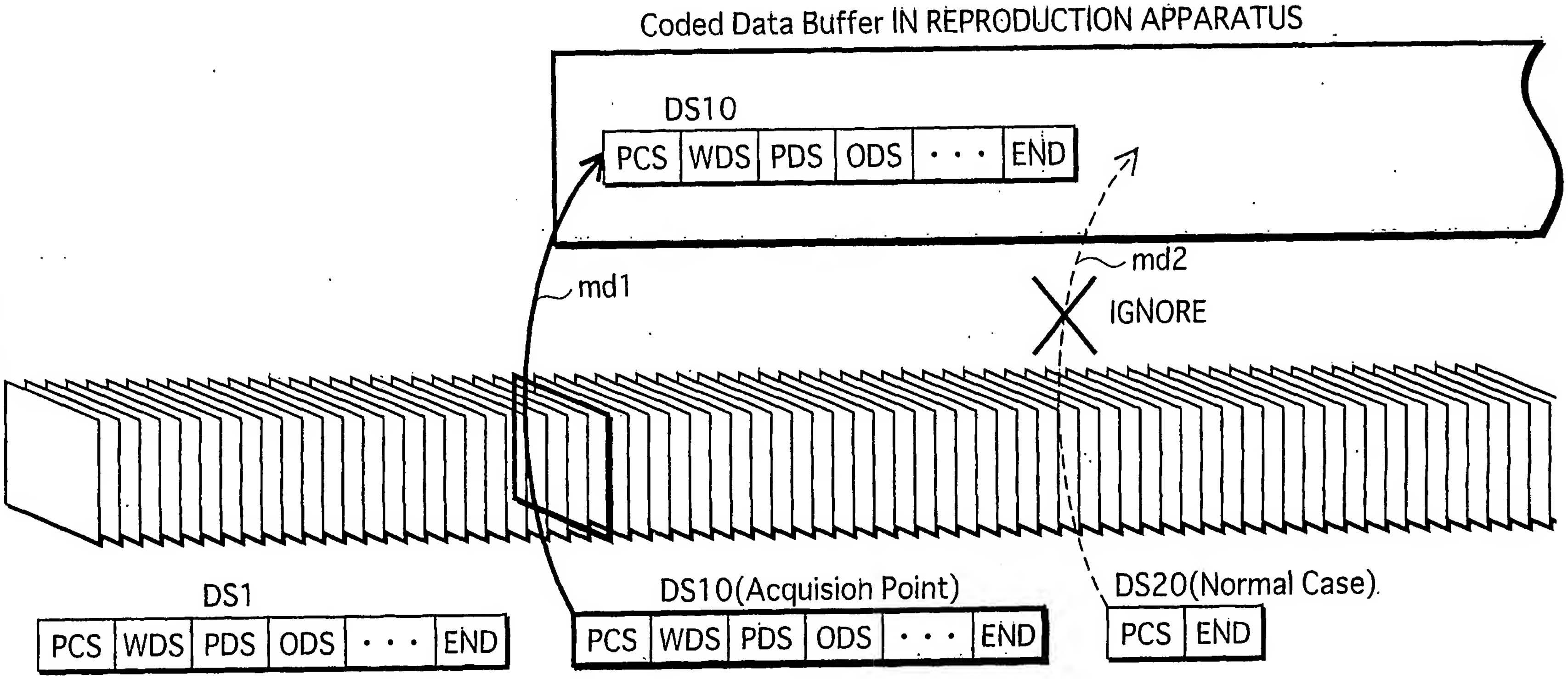

Coded Data Buffer IN REPRODUCTION APPARATUS
DS10
PCS
WDS
PDS
ODS
・・・
END
md1
md2
IGNORE
DS1
PCS
WDS
PDS
ODS
・・・
END
DS10(Acquision Point)
PCS
WDS
PDS
ODS
・・・
END
DS20(Normal Case)
PCS
END

FIG.36

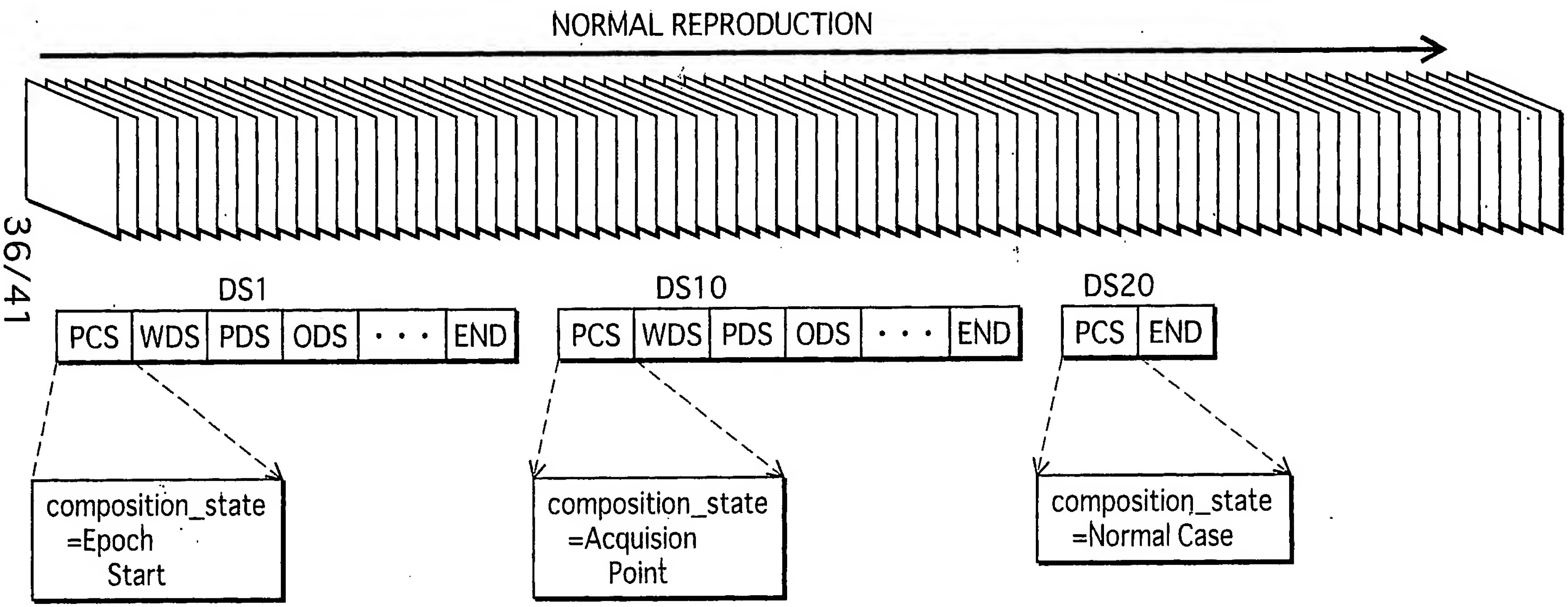

NORMAL REPRODUCTION
DS1
PCS
WDS
PDS
ODS
· · ·
END
DS10
PCS
WDS
PDS
ODS
· · ·
END
DS20
PCS
END
composition_state =Epoch Start
composition_state =Acquision Point
composition_state =Normal Case

FIG.37

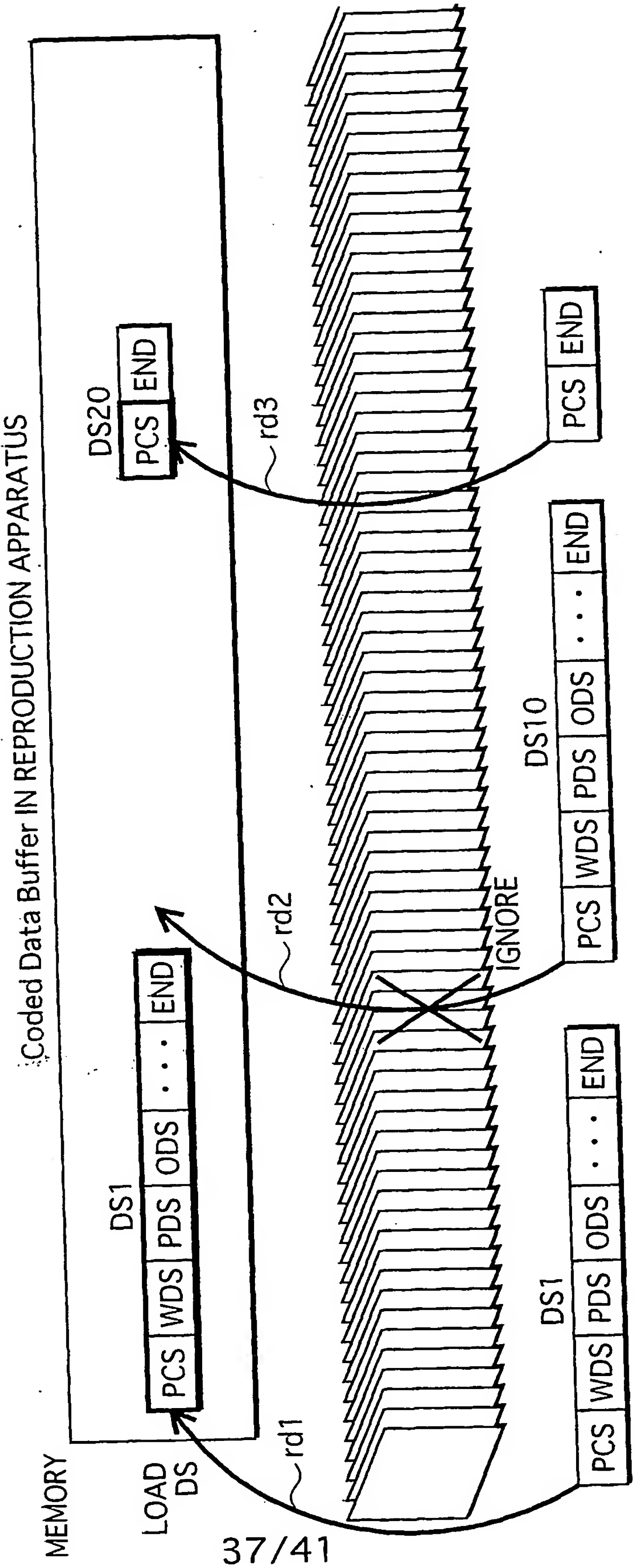

Coded Data Buffer IN REPRODUCTION APPARATUS
MEMORY
LOAD DS
DS1
PCS
WDS
PDS
ODS
• • •
END
DS20
PCS
END
rd1
rd2
rd3
IGNORE
DS1
PCS
WDS
PDS
ODS
• • •
END
DS10
PCS
WDS
PDS
ODS
• • •
END
PCS
END

FIG.38

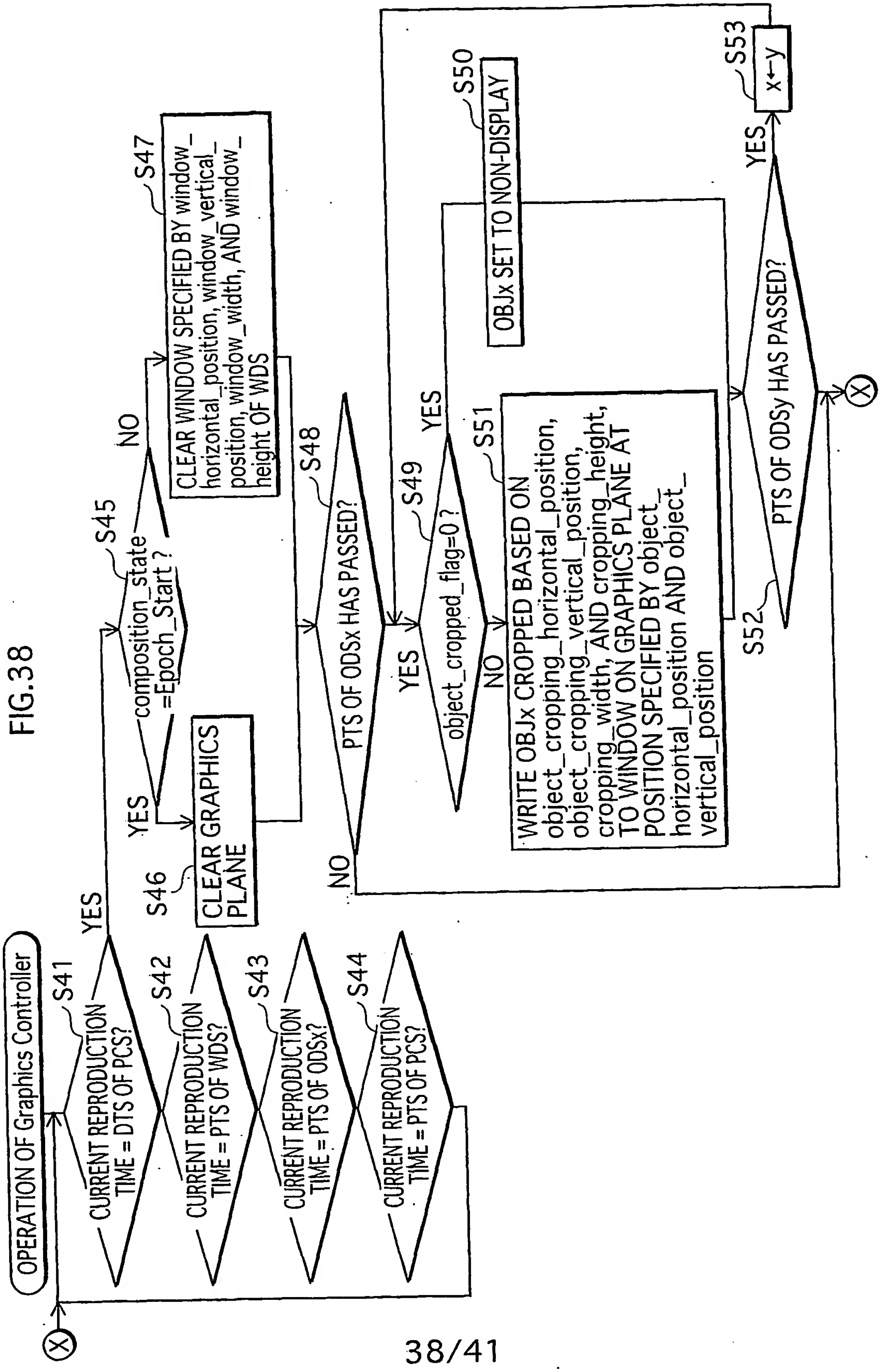

OPERATION OF Graphics Controller
S41
CURRENT REPRODUCTION TIME = DTS OF PCS?
S42
CURRENT REPRODUCTION TIME = PTS OF WDS?
S43
CURRENT REPRODUCTION TIME = PTS OF ODSx?
S44
CURRENT REPRODUCTION TIME = PTS OF PCS?
YES
S45
composition_state =Epoch_Start ?
YES
NO
S46
CLEAR GRAPHICS PLANE
S47
CLEAR WINDOW SPECIFIED BY window_horizontal_position, window_vertical_position, window_width, AND window_height OF WDS
S48
PTS OF ODSx HAS PASSED?
NO
YES
S49
object_cropped_flag=0 ?
YES
NO
S50
OBJx SET TO NON-DISPLAY
S51
WRITE OBJx CROPPED BASED ON object_cropping_horizontal_position, object_cropping_vertical_position, cropping_width, AND cropping_height, TO WINDOW ON GRAPHICS PLANE AT POSITION SPECIFIED BY object_horizontal_position AND object_vertical_position
S52
PTS OF ODSy HAS PASSED?
YES
S53
x←y
X
X

FIG.39

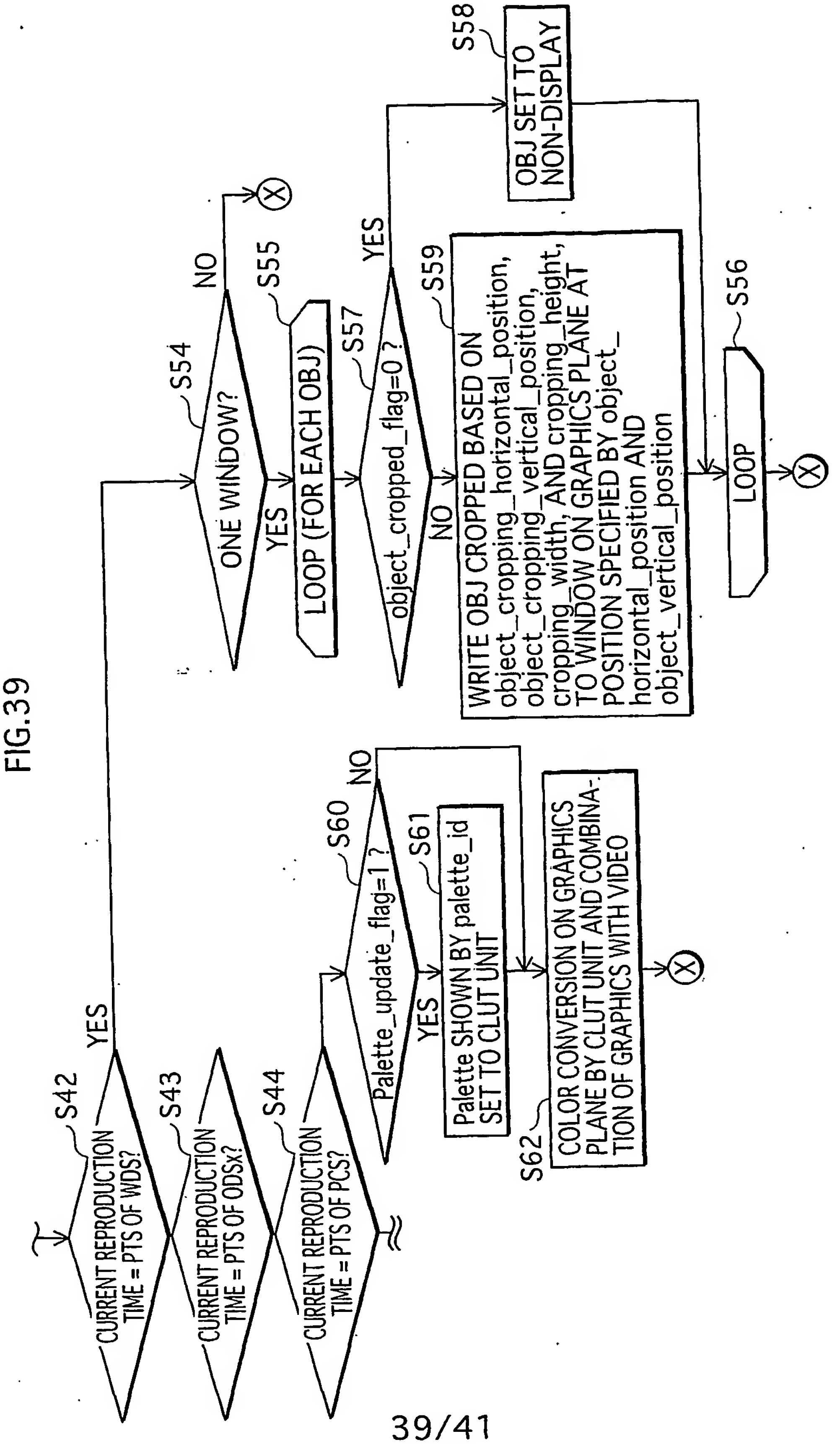

S42
CURRENT REPRODUCTION TIME = PTS OF WDS?
YES
S43
CURRENT REPRODUCTION TIME = PTS OF ODSx?
S44
CURRENT REPRODUCTION TIME = PTS OF PCS?
S54
ONE WINDOW?
YES
NO
Ⓧ
S55
LOOP (FOR EACH OBJ)
S57
object_cropped_flag=0 ?
YES
NO
S58
OBJ SET TO NON-DISPLAY
S59
WRITE OBJ CROPPED BASED ON object_cropping_horizontal_position, object_cropping_vertical_position, cropping_width, AND cropping_height, TO WINDOW ON GRAPHICS PLANE AT POSITION SPECIFIED BY object_horizontal_position AND object_vertical_position
S56
LOOP
Ⓧ
S60
Palette_update_flag=1 ?
YES
NO
S61
Palette SHOWN BY palette_id SET TO CLUT UNIT
S62
COLOR CONVERSION ON GRAPHICS PLANE BY CLUT UNIT AND COMBINATION OF GRAPHICS WITH VIDEO
Ⓧ

FIG.40

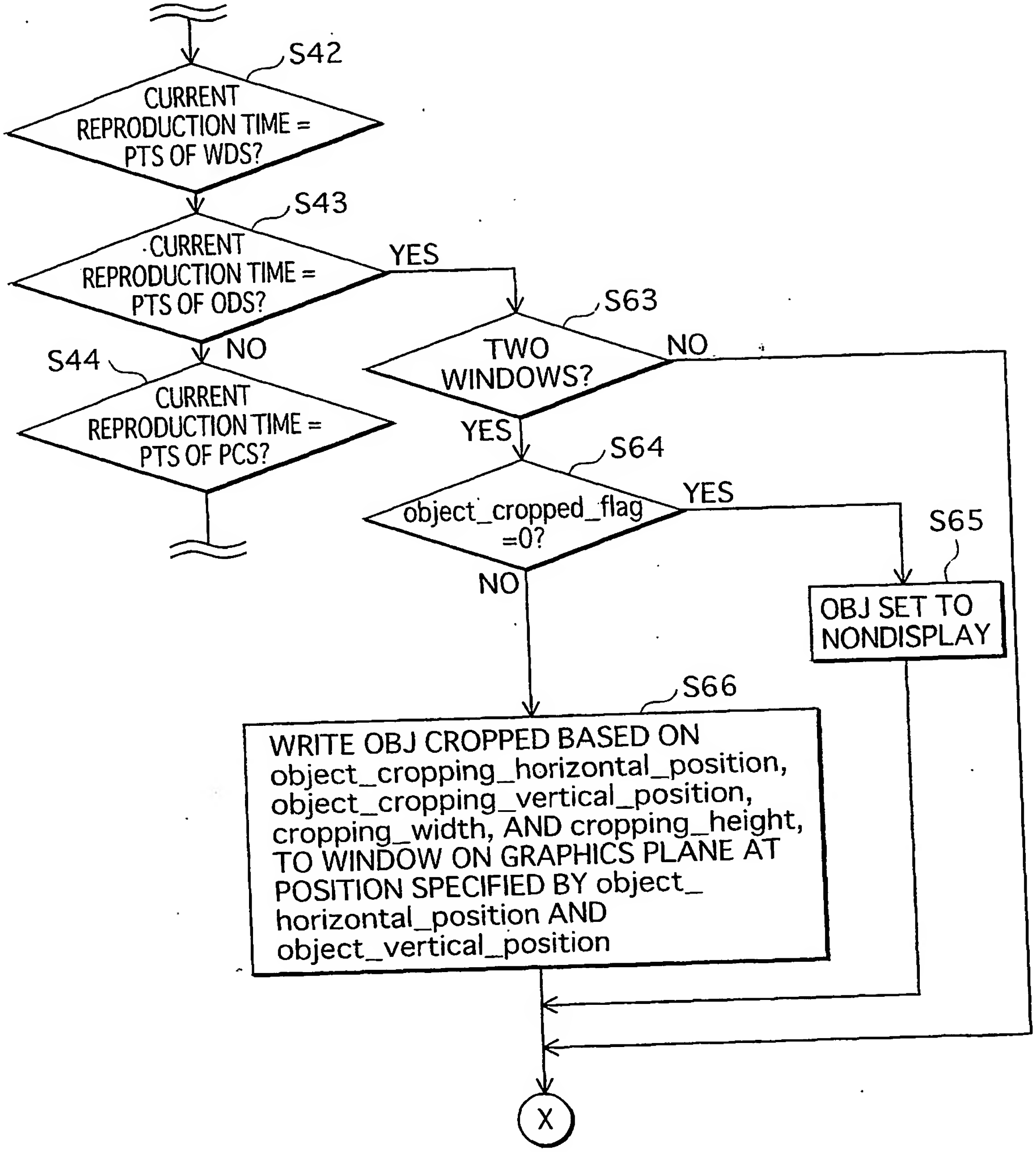
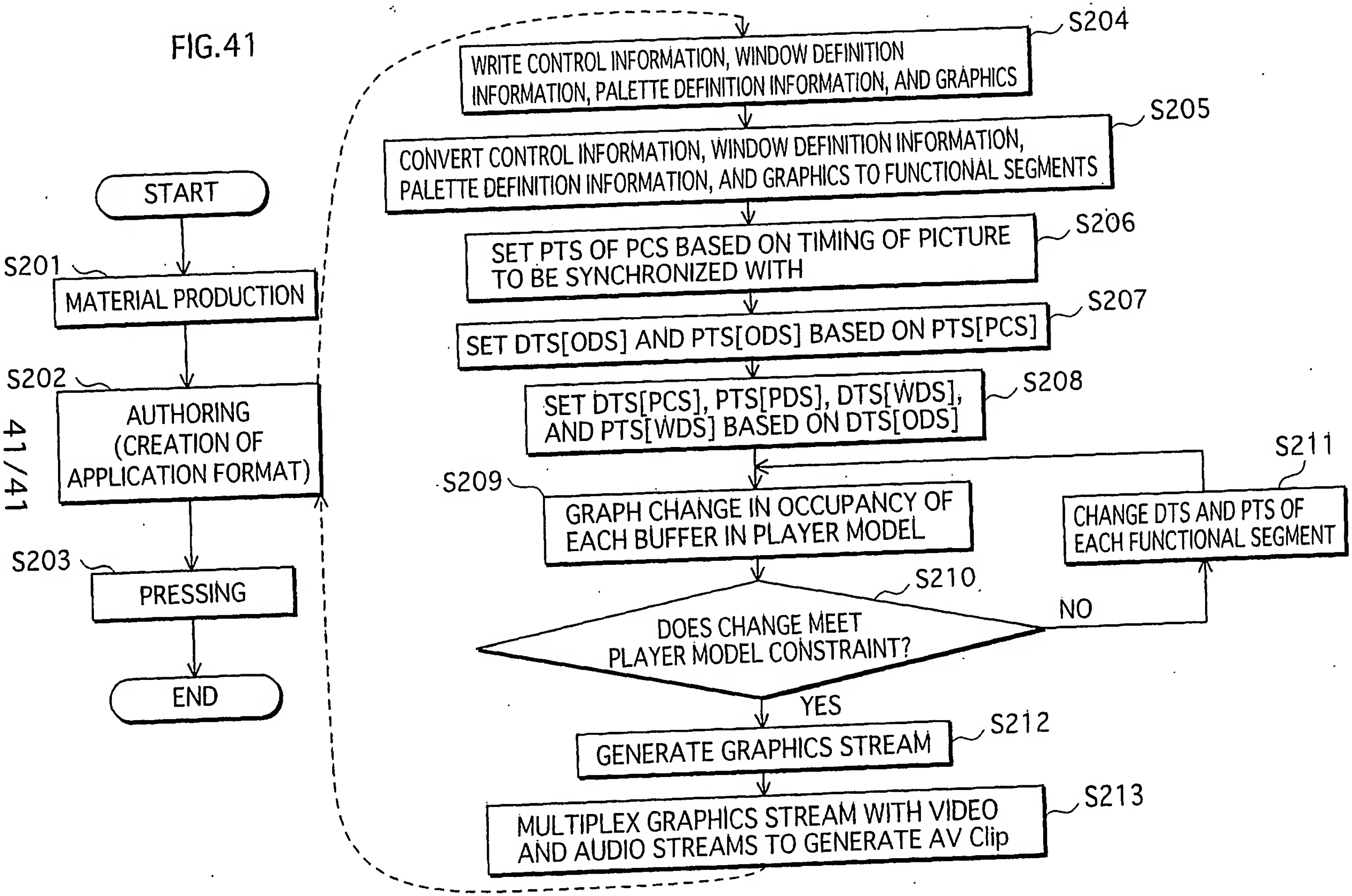

S42
CURRENT REPRODUCTION TIME = PTS OF WDS?
S43
CURRENT REPRODUCTION TIME = PTS OF ODS?
YES
NO
S44
CURRENT REPRODUCTION TIME = PTS OF PCS?
S63
TWO WINDOWS?
NO
YES
S64
object_cropped_flag =0?
YES
NO
S65
OBJ SET TO NONDISPLAY
S66
WRITE OBJ CROPPED BASED ON object_cropping_horizontal_position, object_cropping_vertical_position, cropping_width, AND cropping_height, TO WINDOW ON GRAPHICS PLANE AT POSITION SPECIFIED BY object_horizontal_position AND object_vertical_position
X

FIG.41


START
S201
MATERIAL PRODUCTION
S202
AUTHORING (CREATION OF APPLICATION FORMAT)
S203
PRESSING
END
S204
WRITE CONTROL INFORMATION, WINDOW DEFINITION INFORMATION, PALETTE DEFINITION INFORMATION, AND GRAPHICS
S205
CONVERT CONTROL INFORMATION, WINDOW DEFINITION INFORMATION, PALETTE DEFINITION INFORMATION, AND GRAPHICS TO FUNCTIONAL SEGMENTS
S206
SET PTS OF PCS BASED ON TIMING OF PICTURE TO BE SYNCHRONIZED WITH
S207
SET DTS[ODS] AND PTS[ODS] BASED ON PTS[PCS]
S208
SET DTS[PCS], PTS[PDS], DTS[WDS], AND PTS[WDS] BASED ON DTS[ODS]
S209
GRAPH CHANGE IN OCCUPANCY OF EACH BUFFER IN PLAYER MODEL
S210
DOES CHANGE MEET PLAYER MODEL CONSTRAINT?
NO
S211
CHANGE DTS AND PTS OF EACH FUNCTIONAL SEGMENT
YES
S212
GENERATE GRAPHICS STREAM
S213
MULTIPLEX GRAPHICS STREAM WITH VIDEO AND AUDIO STREAMS TO GENERATE AV Clip